SQLI DIGITAL EXPERIENCE

المندوبية السامية للتخطيط
HAUT-COMMISSARIAT AU PLAN

INSEA

Projet de Fin d'Études

*****

Contribution à la conception et au développement d'une plateforme de commerce électronique pour la marque « Nespresso »

Préparé par :

- **EL HOURI Mohamed El Mehdi**

Sous la direction de :

- Mme. HILAL Imane (INSEA)
- M. XXXX xxxx (SQLI – ISC)

Soutenu publiquement comme exigence partielle en vue de l'obtention du Diplôme d'Ingénieur d'État
Filière : Data & Software Engineering

Devant le jury composé de :

- Mme. HILAL Imane (INSEA)
- M. XXXX xxxx (SQLI – ISC)

- **M. XXXXX Xxxx (SQLI – ISC)**

Juin 2023 / PFE N°

Dédicace

Je voudrais dédier ce travail à toutes les personnes qui ont joué un rôle important dans ma vie, à commencer par mes parents et ma famille qui ont toujours été là pour moi et sans qui je n'aurais aucun sens.

Je voudrais également dédier ce travail aux respectables et honorables professeurs qui m'ont enseigné et soutenu dans le développement de ma personnalité en tant que professionnel compétent.

Je n'oublierai jamais les moments partagés avec mes amis qui ont été mes fidèles compagnons tout au long de ma vie, et j'espère que vous trouverez dans ce travail le témoignage de ma profonde gratitude.

Enfin, je voudrais dédier ce travail à toutes les personnes qui ont participé à son élaboration et à toutes celles que je n'ai pas citées pour leur soutien inconditionnel. Votre contribution a été inestimable et je vous en serai éternellement reconnaissant.

Avec mes remerciements les plus sincères,

Mohamed El Mehdi EL HOURI

Remerciements

En guise de reconnaissance, il me paraît opportun de commencer ce rapport en remerciant ceux qui m'ont beaucoup appris à l'issue de ce travail et ont eu la gentillesse de faire de ce stage un moment très profitable.

Je tiens à remercier dans un premier temps *M. BENNOUNA Youssef*, pour son accueil au sein de l'entreprise et pour m'avoir fait confiance en me donnant la chance de travailler sur un projet d'une telle envergure, ce qui m'a permis de vivre une expérience professionnelle aussi enrichissante.

Je tiens également à exprimer mes sincères remerciements et ma plus haute considération à mon encadrante interne *Mme. HILAL Imane,* qui a bien voulu m'encadrer et me suivre tout au long de ce travail, et aux membres du jury pour m'avoir fait l'honneur d'examiner, d'évaluer et de juger mon travail.

Je ne manquerai pas de remercier l'équipe qui m'a donné la chance de montrer mes compétences à travers ce stage. Mr. XXXX xxxx, Mr. XXXX xxxx et Mr. XXXX xxxx. Grâce à vous, j'ai pu comprendre le cœur du métier, relever une multitude de défis et me dépasser. Je ne pourrai jamais vous remercier d'avoir ouvert la voie de la réussite. Sans votre soutien, et vos conseils, ce travail n'aurait pas été possible.

Enfin, mes plus vives gratifications vont au corps professoral de l'INSEA pour avoir porté un vif intérêt à notre formation, et pour nous avoir accordé le plus clair de leur temps, de leur attention et de leur énergie dans un cadre agréable de complicité et de respect.

Que tous ceux qui m'ont aidé, de près ou de loin, trouvent ici l'expression de mes plus chaleureux remerciements.

Résumé

Ce rapport documente le travail réalisé dans le cadre de mon projet de fin d'études que j'ai effectué au sein de l'entreprise de services numériques SQLI, au cours duquel j'ai contribué à la conception et au développement de la plateforme e-commerce multinationale de la marque Nespresso (NesClub2).

Le projet a marqué la phase initiale du processus de migration depuis l'architecture monolithique existante de NesClub2 vers une architecture basée sur les microservices et a abouti à une application indépendante, capable d'être déployé en dehors de NC2 et s'appuyant entièrement sur des technologies open-source autour de Spring et Kubernetes.

Le projet a obéi à un cycle de vie de développement logiciel personnalisé (SDLC 4.0) basé sur la méthodologie Scrum. Cette approche a permis de garantir une amélioration continue, une certaine agilité et une réactivité à l'égard du client, ce qui a finalement abouti à sa satisfaction.

Mots clés : E-commerce, Nespresso, NesClub2, Microservices, Java EE, Spring, Kubernetes, Cassandra, Scrum, SDLC 4.0

Abstract

This report documents the work carried out as part of my final year project at the digital services company SQLI, during which I contributed to the design and development of Nespresso's multinational e-commerce platform (NesClub2).

The project marked the initial phase of the migration process from the existing monolithic architecture of NesClub2 to a microservices-based architecture and resulted in an independent application, which is deployable outside NC2 and relies fully on open-source technologies around Spring and Kubernetes.

The project adhered to a customized software development lifecycle (SDLC 4.0) which is based on the Scrum methodology. This approach ensured continuous improvement, agility, and responsiveness to the customer, ultimately resulting in its satisfaction.

Keywords : E-commerce, Nespresso, NesClub2, Microservices, Java EE, Spring, Kubernetes, Cassandra, Scrum, SDLC 4.0

Table des matières

INTRODUCTION **1**

CHAPITRE 1 : CONTEXTE GENERAL ET ORGANISME D'ACCUEIL **3**

I. ORGANISME D'ACCUEIL 3
1. Identité et métier 3
2. Services 4
3. Chiffres clés 4
4. Organisation 5
5. ISC Maroc 6
6. Fiche d'identité 7
7. Partenaires 7
8. Clients 7
II. PRESENTATION DU CLIENT 8
III. DESCRIPTION DU PROJET 9
1. Cadre général du projet 9
2. Problématique 9
3. Objectifs du projet 9
IV. METHODOLOGIE ADOPTEE ET PLANIFICATION DU PROJET 10
1. SCRUM 10
2. SDLC 4.0 11
3. Diagramme de GANTT 13

CHAPITRE 2 : ANALYSE ET CONCEPTION **15**

I. ANALYSE DE L'EXISTANT 15
1. Présentation de la plateforme 15
2. Architecture initiale 17
3. Critique de l'existant 17
4. Solution envisagée 18
II. SPECIFICATION DES EXIGENCES 19
1. Identification des acteurs 19
2. Identification des exigences fonctionnelles 20
3. Identification des exigences techniques 21
III. CONCEPTION 22
1. Diagramme de cas d'utilisation 22
2. Conception de l'API 23
3. Diagrammes de séquence 24

CHAPITRE 3 : ENVIRONNEMENT ET OUTILS **29**

I. DEVELOPPEMENT 29
1. Langage : Java 29
2. Framework : Spring / Spring Boot 29
II. TESTS 30
1. JUnit 30
2. Mockito 30
III. CONSTRUCTION DE PROJET 31
1. Apache Maven 31
IV. SGBD 31
1. Apache Cassandra 31
V. IDEs 32
1. Intellij IDEA 32
VI. GESTION DE CODE SOURCE 32
1. Git 32
2. Atlassian Bitbucket 32

VII. DEVOPS 33
1. Docker *33*
2. Jenkins *33*
3. Kubernetes *34*
VIII. DEVELOPPEMENT D'APIS 34
1. Swagger *34*
2. Postman *35*
IX. GESTION DE PROJET 35
1. Atlassian Jira *35*
2. Atlassian Confluence *36*
X. SUIVI DE PERFORMANCE ET REVISION DE CODE 36
1. Prometheus *36*
2. Grafana *36*
3. SonarQube *37*

CHAPITRE 4 : REALISATION **40**

I. ARCHITECTURE TECHNIQUE DU MODULE ECAPI-CARTS 40
II. STRUCTURE DU PROJET 41
1. Domain Driven Design *41*
2. Architecture en oignon *41*
3. Structure multi-modules *42*
III. SCENARIO D'UTILISATION 44
IV. DOCUMENTATION DE L'API 48
V. MONITORING 51

CONCLUSION **54**

Liste des figures

Figure 1: logo SQLI DX 3
Figure 2: expertise SQLI 3
Figure 3: filiales SQLI 5
Figure 4: organisation du groupe SQLI 5
Figure 5 : agences SQLI Maroc 6
Figure 6 : structure ISC Maroc 6
Figure 7 : partenaires SQLI 7
Figure 8: echantillon de clients SQLI 8
Figure 9: logo client Nespresso 8
Figure 10: processus SCRUM 10
Figure 11: processus SDLC 4.0 12
Figure 12: diagramme de Gantt 13
Figure 13: architecture applicative as-is du projet Nespresso 16
Figure 14: architecture initiale de la plateforme e-commerce NC2 17
Figure 15: architecture to-be de la plateforme e-commerce NC2 18
Figure 16: architecture as-is / transitionnelle de la plateforme e-commerce NC2 18
Figure 17: architecture to-be de la plateforme e-commerce NC2 19
Figure 18: diagramme de cas d'utilisation du module eCAPI Cart 22
Figure 19: diagramme de sequence « etablir un devis d'achat pour un client anonyme » 24
Figure 20 : diagramme de sequence « fusionner le panier d'achat lors de l'authentification » 25
Figure 21: diagramme de sequence « synchroniser les donnees depuis NC2 » 26
Figure 22: diagramme de sequence « mettre en cache les donnees » 27
Figure 23 : logo Java 29
Figure 24: logo Spring 29
Figure 25: logo Spring Boot 30
Figure 26: logo JUnit 30
Figure 27: logo Mockito 31
Figure 28: logo Maven 31
Figure 29: logo Apache Cassandra 31
Figure 30: logo Intellij IDEA 32
Figure 31: logo Git 32
Figure 32: logo Bitbucket 32
Figure 33: logo Docker 33
Figure 34: logo Jenkins 34
Figure 35: logo Kubernetes 34
Figure 36: logo Swagger 34
Figure 37: logo Postman 35
Figure 38: logo Jira 35
Figure 39: logo Confluence 36
Figure 40: logo Prometheus 36
Figure 41: logo Grafana 37
Figure 42: logo SonarQube 37
Figure 43: la solution Sonar 38
Figure 44: architecture technique du module ecapi-carts 40
Figure 45: architecture en oignon 41
Figure 46: sous modules du projet ecapi-carts 42
Figure 47: vue de la structure du projet dans Intellij Idea 43
Figure 48: panier d'achat vide 44
Figure 49: page d'accueil de la plateforme Nespresso 44
Figure 50: liste des produits 45
Figure 51: details d'un produit 45
Figure 52: cotation anonyme 46
Figure 53: authentification 46

FIGURE 54: FUSION DU PANIER D'ACHAT 47
FIGURE 55: MISE A JOUR DU PANIER D'ACHAT 48
FIGURE 56: CONTRAT SWAGGER POUR ECAPI-CARTS 49
FIGURE 57: DESCRIPTION DETAILLEE DES DIFFERENTES REPONSES DU PREMIER ENDPOINT 50
FIGURE 58: DESCRIPTION DETAILLEE DES PARAMETRES DU PREMIER ENDPOINT 50
FIGURE 59: NOMBRE D'APPELS AU SERVICE DE COTATION, AFFICHE DANS GRAFANA 51
FIGURE 60: NOMBRE D'APPELS REUSSIS AU SERVICE DE COTATION, AFFICHE DANS GRAFANA 51
FIGURE 61: NOMBRE D'APPELS ECHOUES AU SERVICE DE COTATION, AFFICHE DANS GRAFANA 52

Liste des tableaux

TABLEAU 1: FICHE D'IDENTITE DU GROUPE SQLI [3] 7
TABLEAU 2: ACTEURS ET LEURS ROLES DANS LE SYSTEME 19
TABLEAU 3: CONCEPTION DE L'API 23

Liste des abréviations

ACID	Atomic, Coherent, Isolated, Durable
API	Application Programming Interface
B2B	Business to Business
B2C	Business to Customer
CCO	Chief Operating Officer
CEO	Chief Executive Officer
CMMI	Capability Maturity Model Integrated
CX	Customer Experience
DDD	Domain Driven Design
DTO	Data Transfer Object
DX	Digital Experience
eCAPI	eCommerce API
ESN	Entreprise de services numériques
HTTP	Hypertext Transfer Protocol
IA	Intelligence artificielle
IDE	Integrated Development Environment
ISC	Innovative Service Center
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MAT	Master Acceptance Test
NC2	NesClub2
NCS	Nespresso Corporate Services

NER	NesOA Events Router
NesOA	Nespresso Service Oriented Architecture
NRS	NesSoft Routing Service
REST	REpresentational State Transfer
S. A	Société anonyme
SDLC	Software Development Life Cycle
SSII	Société de services et d'ingénierie en informatique
TDD	Test Driven Development
UAT	User Acceptance Test
URL	Uniform Resource Locator
WIP	Work In Progress

Introduction

Introduction

Toujours en croissance rapide, le commerce électronique est aujourd'hui la méthode d'achat préférée de nombreuses personnes. Les sites e-commerce représentent une part de plus en plus importante des ventes des entreprises bien positionnées sur l'internet. En effet, le commerce électronique offre un certain nombre d'avantages par rapport aux magasins traditionnels, tels que la disponibilité, le prix abordable et une plus grande sélection de produits. Cette course pousse les entreprises à se doter de toutes les dernières innovations technologiques à forte valeur ajoutée et à être le plus compétitif possible sur ce marché.

N'échappant pas à la règle, Nespresso a développé des politiques visant à accroître la performance de son site de commerce électronique, à améliorer la qualité de ses produits et services et à maintenir sa position par rapport à ses concurrents. Ainsi, elle a décidé de lancer un projet de refonte de sa plateforme e-commerce afin d'accompagner la croissance commerciale de sa filiale Nespresso à travers le monde.

Le travail effectué dans le cadre de ce projet fait partie de la maintenance adaptative de la plateforme internationale de commerce électronique du projet Nespresso adoptée par SQLI. L'objectif de ce dernier est d'analyser, de concevoir, de développer et de surveiller les performances de la plateforme, conformément à l'engagement de SQLI de fournir des applications fiables et performantes au client.

La plateforme e-commerce monolithique de Nespresso n'était plus en mesure de répondre à ses besoins au fur et à mesure qu'elle augmentait en complexité. Par conséquent, l'entreprise a décidé d'adopter une architecture en microservices. Nesspresso espère que cette décision lui permettra d'améliorer l'évolutivité, la maintenabilité et la flexibilité de sa plateforme, ce qui l'aidera à augmenter ses profits et à mieux satisfaire ses clients.

Afin de présenter le travail qui a abouti à la concrétisation de la solution proposée, ce rapport est organisé en quatre chapitres :

- Le premier chapitre introduira d'abord SQLI en tant que structure d'accueil et fournira ensuite une vue d'ensemble du projet, y compris le problème qu'il cherche à résoudre, ses objectifs et la méthodologie de gestion de projet qui sera suivie pour y parvenir ;
- Le deuxième chapitre sera consacré à une critique détaillée du système existant, à l'analyse des besoins fonctionnels et techniques de la solution cible et à la modélisation graphique de ces besoins à l'aide d'UML ;
- Le troisième chapitre présentera l'environnement et les outils choisis pour le développement du projet ;
- Le dernier chapitre abordera la réalisation du projet, en donnant un aperçu de la nouvelle architecture de la solution mise en œuvre et de certains scénarios de son utilisation.

Chapitre 1 : contexte général et organisme d’accueil

Chapitre 1 : contexte général et organisme d'accueil

Ce chapitre introductif a pour objectif d'exposer le cadre général dans lequel se déroule le projet, en présentant en premier lieu l'organisme d'accueil, son histoire, l'ensemble de ses activités, ses domaines d'expertise et ses clients. Il vise, en deuxième lieu, à décrire le projet et son cadre fonctionnel, la problématique à résoudre et les objectifs du projet. La dernière section est consacrée à la présentation de la démarche adoptée par l'équipe de travail pour assurer le bon déroulement du projet et parvenir à ses objectifs.

I. Organisme d'accueil

1. Identité et métier

Créé en 1990, le Groupe **SQLI** est le leader français des sociétés de services spécialisées dans les nouvelles technologies de l'information et de la communication. L'entreprise se positionne en tant que "Grand Spécialiste" en proposant une offre globale alliant la capacité de production industrielle d'un groupe international à l'expertise et la souplesse informatique d'un spécialiste, en mesure de donner de la valeur aux innovations des entreprises qu'elle accompagne. [1]

Figure 1: logo SQLI DX

SQLI s'est spécialisée dans les projets e-commerce, liés aux systèmes d'informations intégrant l'utilisation des technologies internet. Pour aider les entreprises à tirer profit des technologies Internet, SQLI propose un accompagnement global sur tout le cycle du projet :

- Des prestations de conseil pour aider les clients à faire les bons choix ;
- La mise en œuvre concrète de ces choix par la réalisation ;
- L'intégration et l'accompagnement dans le déploiement des projets et le transfert de compétences.

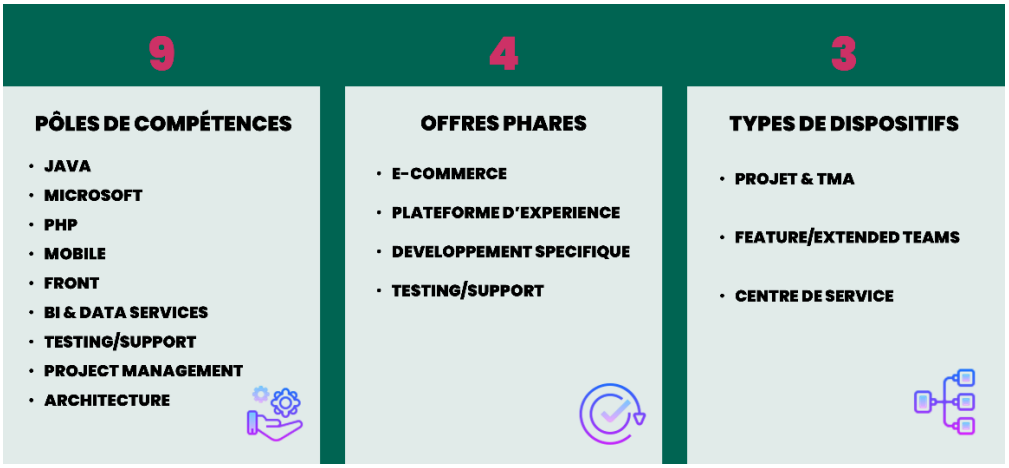


Figure 2: expertise SQLI

2. Services

a. Technology & Transformation

SQLI crée des environnements technologiques pour des plateformes évolutives et des solutions sur-mesure en adaptant des méthodologies facilitatrices pour imaginer, structurer et mettre en œuvre la transformation digitale.

b. Experience Platforms

Le groupe construit des ensembles de solutions adaptées à l'écosystème, aux exigences et aux objectifs de performance des entreprises afin de leur donner le pouvoir de créer, délivrer et optimiser des expériences digitales riches et personnalisées, au service des clients, des employés et des partenaires.

c. E-Commerce

SQLI aide les entreprises à se démarquer et accroître leur chiffre d'affaires en combinant le meilleur de la stratégie, du design et de la technologie. Que ce soit dans un environnement BtoB ou BtoC, nous concevons des expériences holistiques centrées sur l'émotion et la valeur à chaque étape du parcours utilisateur.

d. Digital Consulting

Aider les équipes de direction, les services opérationnels et les départements IT à s'aligner sur une vision innovante commune tirant parti des opportunités offertes par le digital.

e. Data & Insights

Le groupe aide les entreprises à définir leur vision et leur stratégie en matière de données, de la prise de décision à l'analyse prédictive en leur fournissant la plateforme et les outils qui leur permettront d'innover, d'améliorer leur efficacité opérationnelle, ou mieux connaître et comprendre leurs clients.

f. Digital Marketing & Design

Aider les organisations à concevoir des stratégies digitales moteurs de croissance et de visibilité, avec l'utilisation de méthodologies de Design Thinking pour générer de la notoriété et faciliter la conversion, dans une logique de fidélisation des clients. [2]

3. Chiffres clés

Les chiffres clés suivants présentent la situation actuelle de SQLI :

- **30** ans d'expérience et d'innovation : SQLI place son développement sur une expertise technologique de pointe et sur sa politique intense de veille. En 2005, SQLI devient la première SSII française à obtenir la certification CMMI 3 ;
- Plus de **2200** collaborateurs au groupe SQLI : dont plus de 950 experts du digital et plus de 500 certifications IT (Scrum, Java, Microsoft …) ;
- **246 M€** de chiffre d'affaires en 2022 : Le groupe SQLI confirme son retour à la croissance grâce à une offre en adéquation avec les attentes du marché et une reprise progressive de la demande en Services Informatiques. [1]

4. Organisation

Le groupe est organisée sous forme de 35 agences de proximité dans 13 pays et de plus de 2200 collaborateurs, afin de conserver le maximum de réactivité face aux besoins de ses clients. SQLI offre une approche sur mesure aux enjeux spécifiques des entreprises.

SQLI est présente en France, Suisse, Luxembourg, Maroc (Rabat, Casablanca et Oujda), en Espagne, en Belgique et au Canada, Singapour et Suède.

Figure 3: filiales SQLI

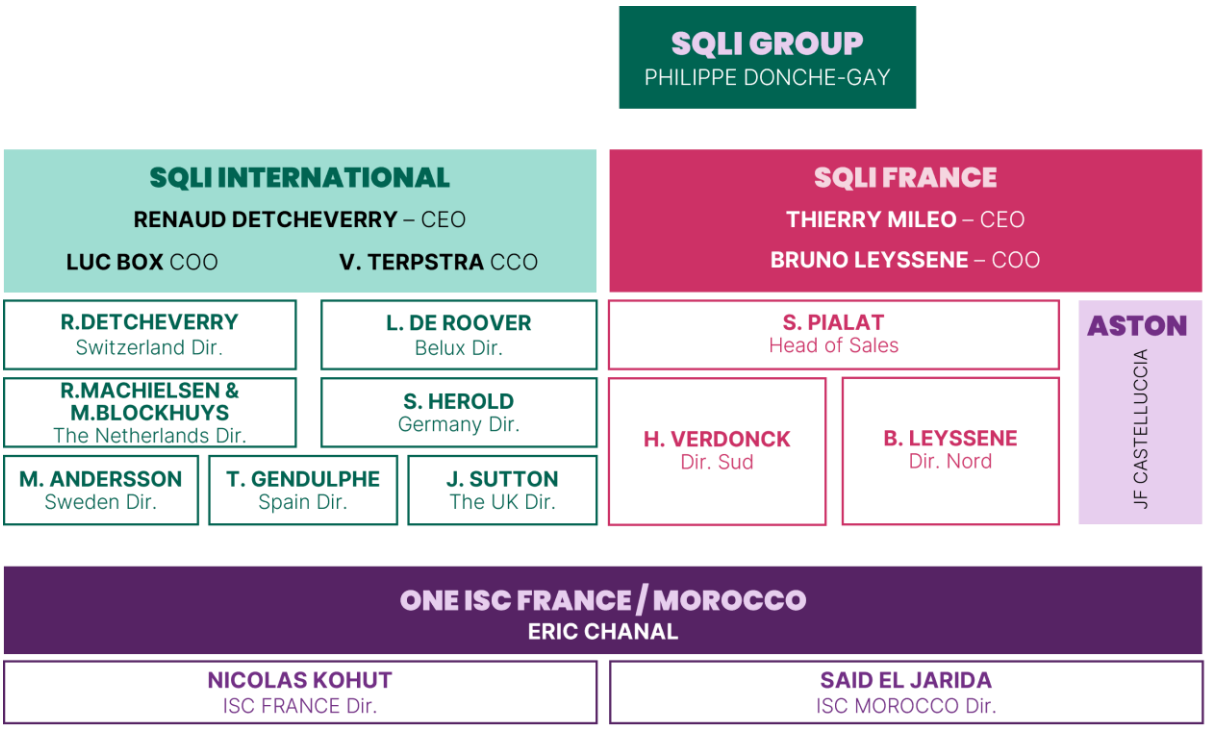


Figure 4: organisation du groupe SQLI

5. ISC Maroc

Fondé en 2003 à Rabat, SQLI Maroc est le centre de Delivery et d'Innovation du Groupe SQLI. Implanté sur 3 sites (Rabat, Oujda et Casablanca), il s'agit d'une organisation au service des client qui met en œuvre et déploie des solutions au niveau global. [1]

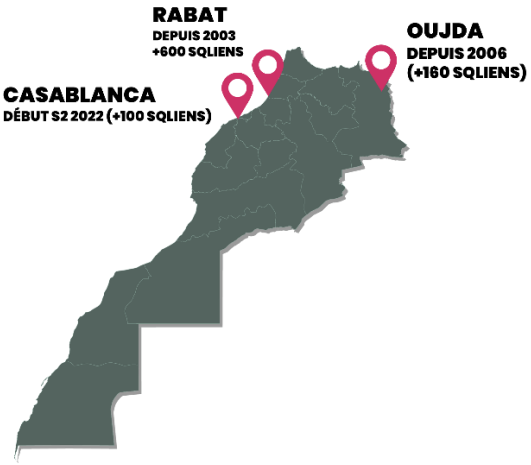


Figure 5 : agences SQLI Maroc

Ces centres de services digitaux sont au cœur du dispositif SQLI et supportent toutes les offres SQLI permettant de répondre avec flexibilité et agilité à tous les clients SQLI.

Les principaux avantages compétitifs des ISC étant :

- Une maîtrise du modèle reposant sur plus de 20 ans d'expérience : 1er centre offshore fin 2003 ;
- Une forte expertise dans les technologies du digitale et des solutions de commerce unifié ;
- Une organisation adaptée aux projets multi-sites et aux fluctuations du volume d'activité ;
- Une livraison pilotée à base de KPI et en transparence avec le client.

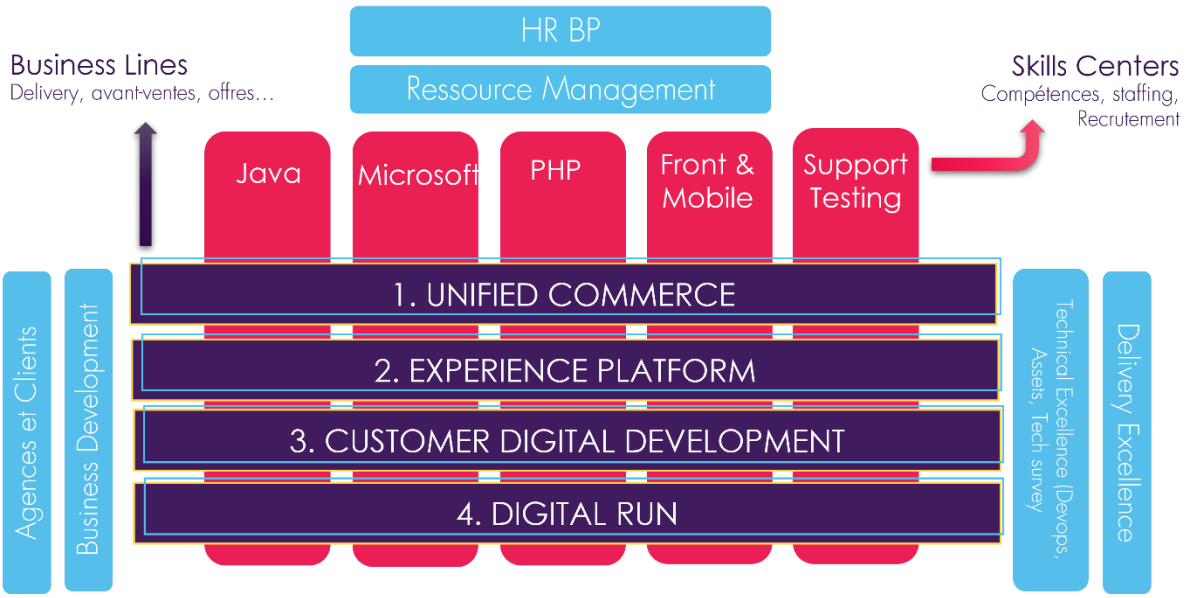


Figure 6 : structure ISC Maroc

6. Fiche d'identité

Création	28-03-1990
Fondateurs	Alain Lefebvre et Jean Rouveyrol
Forme juridique	Société anonyme à directoire au capital
Siège social	166 rue Jules Guesde - 92300 Levallois-Perret France
Direction	Philippe Donche-Gay, CEO depuis 2020
Activité	Conseil en systèmes et logiciels informatiques
Produits	Conseil en transformation digitale, logiciels
SIREN	353861909
Site web	www.sqli.com
Chiffre d'affaires	246 M€ (2022)

Tableau 1: fiche d'identité du groupe SQLI [3]

7. Partenaires

Le positionnement de SQLI à la conjonction du monde numérique et du système d'information de l'entreprise est sa valeur ajoutée. Le groupe a noué un réseau de partenaires pour répondre à ses enjeux de la manière la plus efficace possible, et apporter le meilleur conseil, la meilleure solution.

Figure 7 : partenaires SQLI

8. Clients

Le groupe SQLI accompagne les plus grandes marques dans la conception et la mise en œuvre d'expériences et de dispositifs digitaux. Le groupe travaille pour la quasi-totalité des secteurs d'activité sur des sujets aussi variés que le développement de plateformes d'expérience, des programmes d'agilité à l'échelle ou encore des solutions d'IA, ainsi que pour 3 des 5 plus grands groupes de luxe sur des projets E-commerce d'envergure.

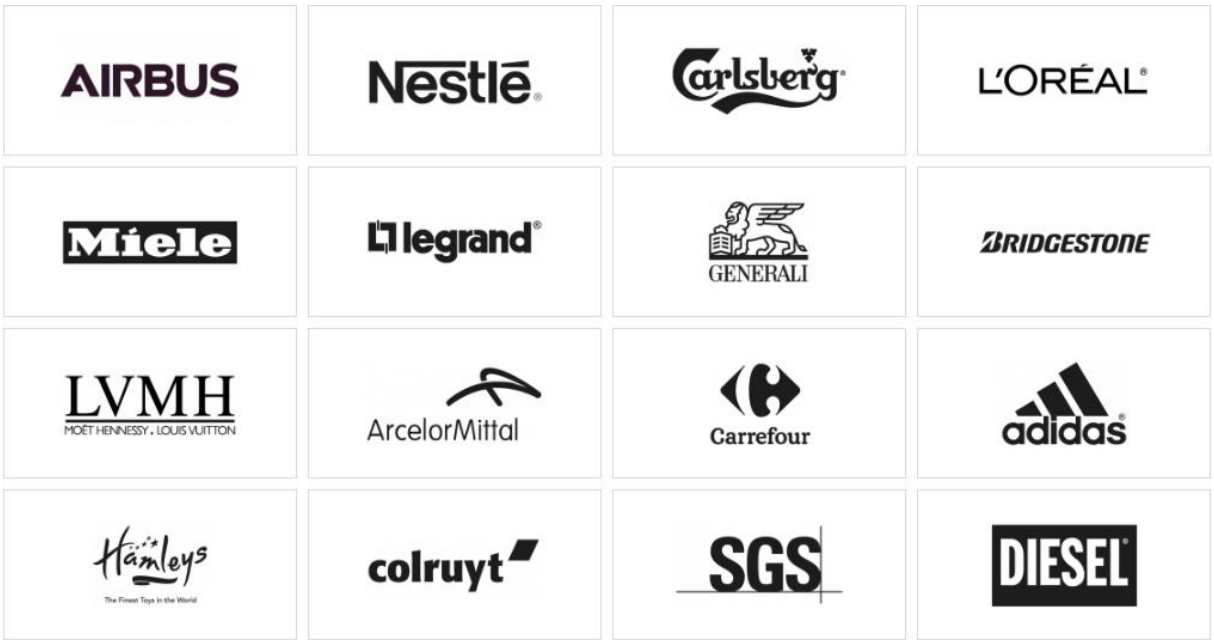


Figure 8: échantillon de clients SQLI

II. Présentation du client

Nespresso, ou Nestlé Nespresso S.A., dont le siège est situé à Lausanne, en Suisse, est une filiale du groupe Nestlé. Positionnée sur le marché du café en dosette de luxe, elle propose des machines haute technologie utilisant des capsules en aluminium.

La marque est un mot-valise constitué à partir de **Nes**tlé et d'Ex**presso**. [4]

Figure 9: logo client Nespresso

Nespresso est présent dans 62 pays et emploie plus que 10 500 employés, en 2022 l'entreprise a généré plus de 7 milliards de dollars de revenus.

III. Description du projet

1. Cadre général du projet

Le travail réalisé s'inscrit dans le cadre de la maintenance adaptative de la plateforme e-commerce internationale (**NC2**) du projet Nespresso adopté par SQLI, Ce dernier vise à suivre l'analyse, la conception, le développement ainsi que le suivi de performance afin d'honorer l'engagement de SQLI à livrer des applications fiables et de haute performance au client.

La plateforme appuie de nombreux pays avec des sites web e-commerce spécifiques et offre une gamme de fonctionnalités et de ressources pour soutenir les différents aspects des opérations de vente en ligne aux clients individuels de Nespresso (B2C) ainsi qu'à ses clients professionnels (B2B).

2. Problématique

En e-commerce, les entreprises cherchent constamment à accroître leur part de marché en offrant des services simples et efficaces qui répondent aux aspirations des clients. Nespresso ne fait pas exception. L'entreprise cherche toujours à fidéliser ses clients en améliorant ses services et ses promotions, et bien sûr à gagner de nouveaux membres en leur proposant des fonctionnalités plus attrayantes.

Cependant, la plateforme initiale de Nespresso (NC2) reposait sur une structure monolithique, ce qui compliquait les changements. Les bugs étaient corrigés et déployés au prix d'efforts considérables et inutiles, et la défaillance d'un élément entraînait des répercussions importantes sur l'ensemble de la plateforme. De plus, la base de données, particulièrement volumineuse, était accessible par tous les modules en même temps, ce qui ralentissait les temps de réponse, surtout en cas de montée en charge.

Enfin, bien que NesClub2 soit entièrement basé sur Hybris et ait longtemps bénéficié de ses capacités robustes, au fur et à mesure du développement de l'application, Nespresso a rencontré des défis, principalement liés aux réglementations imposées par SAP.

3. Objectifs du projet

Face à ces défis, SQLI a posé comme objectifs de ce projet :

- **Indépendance opérationnelle** : les composants doivent pouvoir être étendus, libérés, déployés et exploités indépendamment l'un de l'autre, à tout moment, sans avoir d'incidence sur les autres composants ni entraîner de temps d'arrêt important.
- **Appropriation par l'équipe** : la taille du service est suffisamment petite pour que l'équipe dispose d'une autonomie et d'une propriété totales sur le service, de la définition et la conception à la production, déploiement et maintenance.
- **Prise en charge des pics de charge** : le système doit être capable de ne faire évoluer que les composants qui sont sollicités en cas de pic de charge.
- **Tolérance aux pannes** : la journalisation et le traçage des services entre les composants sont utilisés pour monitorer et diagnostiquer les problèmes.
- **Découplage par rapport à Hybris** : toutes les dépendances à l'égard de Hybris doivent être éliminées afin en mesure de mettre Hybris hors service à terme.

IV. Méthodologie adoptée et planification du projet

1. SCRUM

Afin d'assurer une bonne conduite du projet, la méthodologie agile "SCRUM" est adoptée en raison du fait qu'elle permet à l'équipe de développement de progresser graduellement, de s'ajuster au fur et à mesure aux nouvelles exigences, d'impliquer le client et d'échanger bien davantage avec lui.

Scrum est un cadre léger, itératif et incrémental pour le développement, la livraison et le maintien de produits complexes. Il est conçu pour des équipes de dix membres ou moins qui répartissent leur travail en objectifs susceptibles d'être atteints au cours d'itérations limitées dans le temps, appelées « sprints ».

Chaque sprint ne dépasse pas un mois et dure généralement deux semaines. L'équipe scrum évalue le progrès lors de réunions quotidiennes limitées dans le temps, d'une durée maximale de 15 minutes, appelées « daily scrums ». À la fin du sprint, l'équipe tient deux autres réunions : une revue de sprint « sprint review » destinée à présenter le travail effectué aux parties prenantes et à solliciter des feedbacks, et une rétrospective de sprint « sprint retrospective » destinée à permettre à l'équipe de de faire le point sur ses efforts de collaboration et d'améliorer son travail.

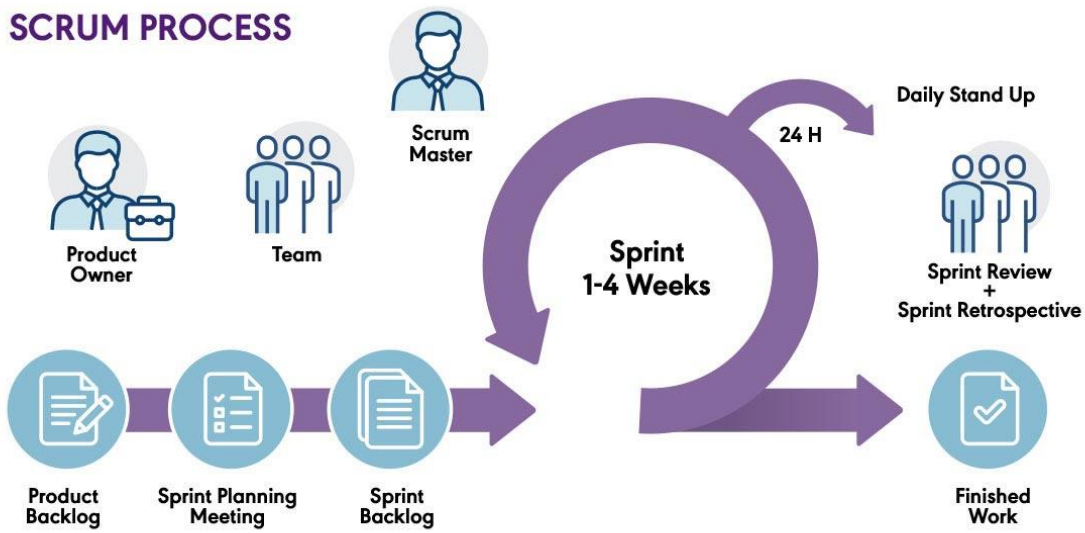


Figure 10: processus SCRUM

La méthodologie SCRUM fait intervenir 3 rôles principaux :

- Les développeurs (**Scrum Team**) : s'engagent à créer n'importe quel aspect d'un incrément utilisable à chaque sprint. Leur rôle est de livrer un incrément de produit ;
- Le **Product Owner :** responsable de la maximisation de la valeur du produit résultant du travail des développeurs. Il fournit de la valeur et gère le « Product Backlog » ;

- Le **Scrum Master** : responsable de la mise en place de Scrum tel que défini dans le Scrum Guide au sein de l'équipe et de l'organisation. Il facilite la mise en œuvre de Scrum.

On distingue plusieurs artefacts scrum :

- **Product Backlog** : liste émergente et ordonnée de ce qui est nécessaire pour améliorer le produit ;
- **Product Goal** : décrit l'état futur du produit qui peut servir de cible à l'équipe Scrum pour planifier ;
- **Sprint Goal** : est un plan par et pour les développeurs sur le travail qu'ils prévoient d'accomplir pendant le Sprint pour atteindre l'objectif de Sprint ;
- **Product Increment** : est une étape concrète vers l'objectif de produit. Il ne s'agit pas seulement de ce que l'équipe fait au cours du dernier sprint, mais plutôt de l'ensemble du produit ;
- **Definition of Done :** est une description formelle de l'état de l'incrément lorsqu'il répond aux mesures de qualité requises pour le produit ;
- **Definition of Ready :** est une description formelle des critères et des conditions qu'une tâche ou une fonctionnalité doit satisfaire avant d'être considérée comme prête à être réalisée ;
- **User Story** : est une explication non formelle, générale d'une fonctionnalité logicielle écrite du point de vue de l'utilisateur final ;
- **Epic** : un travail de grande envergure qui peut être décomposé en un certain nombre de petites user stories. Les epics sont presque toujours livrées au cours d'une série de sprints. [5]

2. SDLC 4.0

Sur la base SCRUM, le projet Nespresso obéit à un cycle de vie de développement logiciel personnalisé (SDLC 4.0) qui vise à :

- **Réduire le décalage entre les équipes agiles et les équipes de livraison :** en ayant un seul SDLC comme référence et le même ensemble d'outils disponibles ;
- **Renforcer le principe de « User Centricity » :** en veillant à ce que les user stories soient rédigées par les Business Analysts ;
- **Réduire le WIP :** en ne clôturant les user stories que lorsque toutes les activités de test liées à la validation de la fonctionnalité sont achevées ;
- **Adopter une approche de « Shift Left Testing » :** en encourageant les tests d'acceptance par l'utilisateur (UAT) dans les environnements d'intégration, dès que les Epics sont terminés.

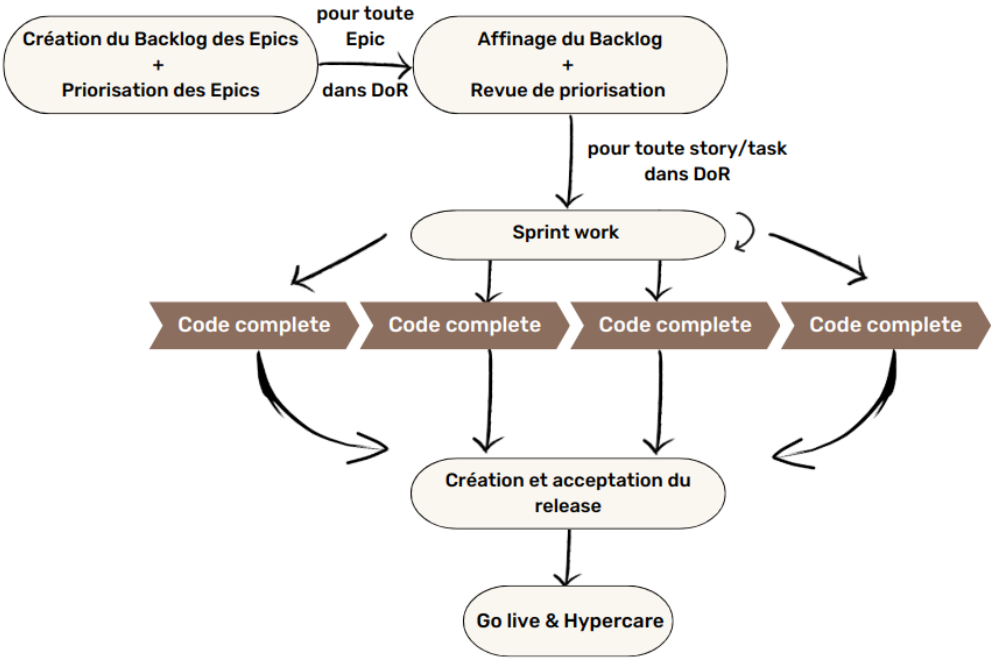


Figure 11: processus SDLC 4.0

Le processus SDLC 4.0 comprend les tâches suivantes :

a. Création du backlog des epics et priorisation des epics

- Définition initiale d'une fonctionnalité (changement fonctionnel ou technique) ;
- Création de l'epic ;
- Priorisation de l'epic.

b. Raffinement du backlog et révision des priorités

- Découpage de l'epic en user stories et tâches ;
- Révision des priorités pour les user stories et les tâches divisées dans le Product Backlog.

c. Sprint Work

- **Sprint Planning** : étape de planification visant à établir les objectifs de développement du sprint sur la base d'un examen du Product Backlog.
- **Sprint Execution** : consiste à s'auto-organiser pour accomplir le travail, en se focalisant sur la clôture des tâches individuelles et en veillant à ce que les critères d'acceptation soient respectés.
- **Sprint Review** : effectuée à la fin pour présenter les résultats aux parties prenantes et recueillir leurs commentaires afin d'ajuster le Product Backlog.
- **Sprint Retrospective** : organisée pour refléter la réussite du sprint et planifier des améliorations pour l'avenir.

d. Création et acceptation de la livraison

Pour qu'un Epic soit inclus dans la release, il doit avoir le statut "In Acceptance" (développement + tests + documentation …). Les seules tâches en suspens doivent être celles liées à la validation de la fonctionnalité dans l'environnement de staging, UAT, MAT et les tâches liées au déploiement dans l'environnement de staging/préproduction/production.

e. Go Live and Hypercare

- Demande de changement, y compris la création d'un ticket, la mise à l'essai et le déploiement en production ;
- Hypercare : vérification du bon fonctionnement de la fonctionnalité fournie et de la mise en place de toutes les métriques et alarmes ;
- Fermeture de l'epic.

L'événement "Code Complete" est un jalon indiquant que toutes les tâches et fonctionnalités de l'epic sont terminées et que le code source est prêt pour l'intégration, les tests et le déploiement.

3. Diagramme de GANTT

Le diagramme de Gantt ci-dessous offre une représentation visuelle claire de la planification et de la progression réelle du projet sur la période qui s'étend du 1er mars au 26 mai 2023.

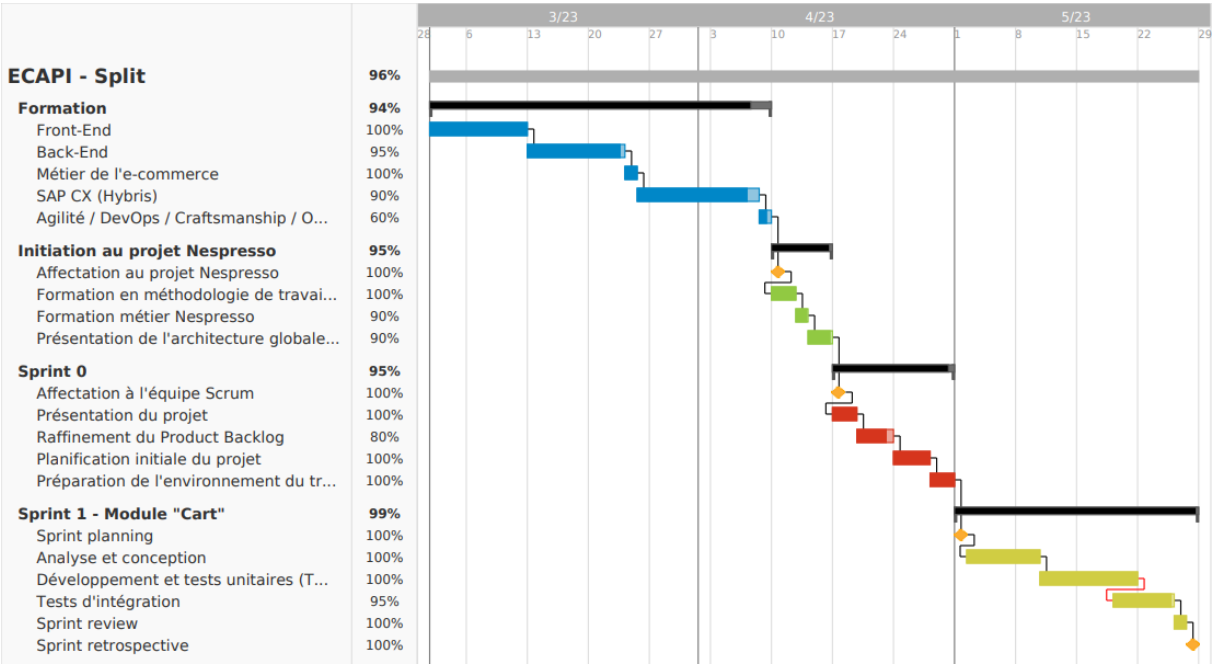


Figure 12: diagramme de Gantt

Ce premier chapitre a posé les bases du projet en en exposant son contexte général, les objectifs fixés pour relever ses défis, et le processus de développement adopté pour sa réalisation. Le chapitre suivant fournira un aperçu détaillé du processus d'analyse et de conception mené pour élaborer des solutions conformes aux objectifs définis.

Analyse et conception

Chapitre 2 : Analyse et conception

Le présent chapitre s'articule autour de trois principaux volets :

- Un premier volet qui analyse le système existant et présente les restrictions de son architecture ;
- Un deuxième volet qui définit les exigences fonctionnelles et techniques auxquelles l'application cible doit répondre ;
- Et un troisième volet qui utilise UML pour modéliser et représenter graphiquement les fonctionnalités formulées auparavant.

I. Analyse de l'existant

1. Présentation de la plateforme

Le projet Nespresso a connu plusieurs versions et continue d'évoluer. La solution actuelle comprend, en plus de l'application web (**NC2**), un réseau middleware (**NesOA**) dédié à la communication entre la plateforme de commerce électronique et l'ERP Nespresso (**NesSoft**) qui fournit la majorité des données pour les sites web.

a. NC2

NC2 est le principal monolithe, ses principales fonctionnalités incluent :

- La gestion et affichage du contenu marketing ;
- L'affichage du catalogue de produits, doté de mécanismes de commande rapide, de filtrage et de comparaison ;
- La gestion du panier d'achat ;
- La gestion du processus d'achat en ligne, avec plusieurs modes de livraison et de paiement ;
- La personnalisation de l'expérience des utilisateurs (navigation et contenu personnalisés en fonction des groupes cibles comportementaux).

b. NesSoft

NesSoft est le progiciel de Nespresso, il gère presque tout : les données clients, les produits, les commandes, les devis, la logique commerciale, la logistique, les fournisseurs, l'entreposage, les finances, ...

c. NCS-Retail

Il s'agit d'un middleware permettant de se connecter à Nessoft. Il est principalement utilisé par NC2 mais aussi par d'autres clients.

- NCS retail a sa propre base de données de cache ;
- La plupart des données sont mises en cache et synchronisées en lots avec Nessoft, à l'aide de cronjobs ;
- Certaines données comme les devis ou les données clients ne sont pas mises en cache dans la base de données, elles sont invoquées en temps réel depuis Nessoft lorsqu'elles sont demandées par NC2.

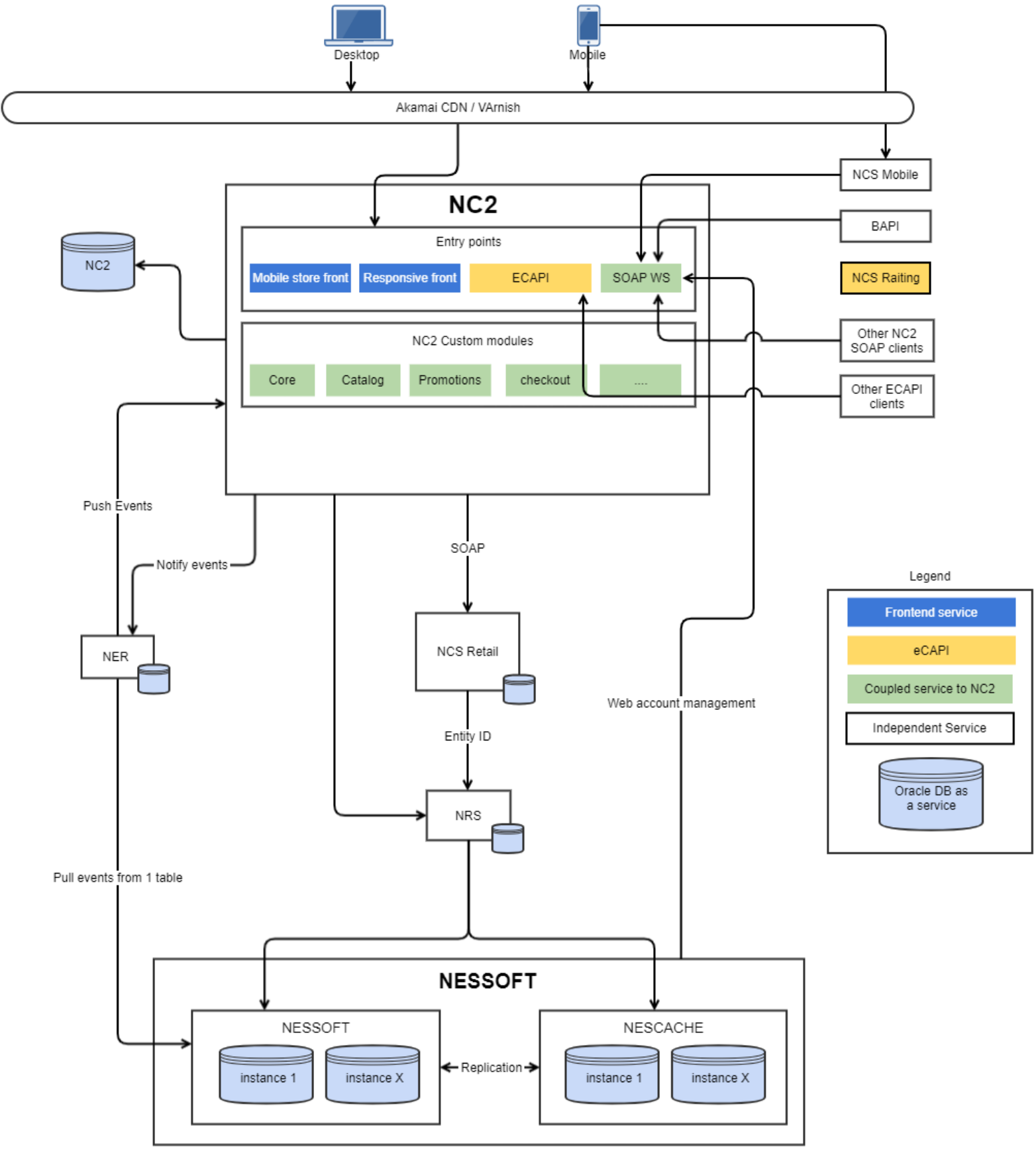


Figure 13: architecture applicative as-is du projet Nespresso

2. Architecture initiale

L'architecture initiale de la solution NC2 consistait en une API basée sur SAP CX (Hybris), exposant les fonctionnalités de base de la plateforme Hybris en tant que « business services » à ses clients.

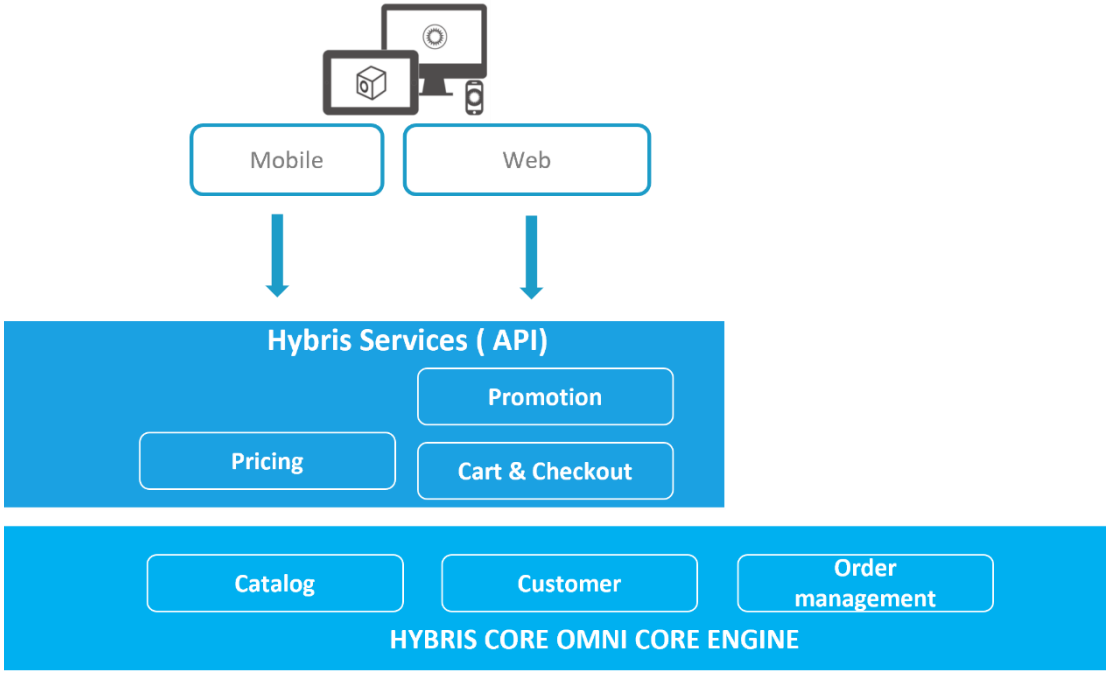


Figure 14: architecture initiale de la plateforme e-commerce NC2

3. Critique de l'existant

En dépit de toutes les fonctionnalités «out of the box » offertes par la technologie de commerce électronique (Hybris), son exploitation présente plusieurs enjeux pour Nespresso :

a. Coût

SAP CX est un logiciel commercial qui peut être assez coûteux. Outre les frais de licence, il y a des frais supplémentaires pour la maintenance, les mises à jour, la personnalisation et la formation.

b. Dépendance vis-à-vis de SAP

Hybris est une solution propriétaire développée par SAP, ce qui signifie qu'il est étroitement lié à la stratégie de SAP et est affecté par les changements de cette dernière. D'une autre part, si le client est dépendant de la plateforme et de ses fonctionnalités, cela limite sa capacité à innover ou à adopter de nouvelles technologies.

c. Scalabilité

La nature monolithique de la solution de commerce électronique rend difficile son évolution horizontale pour répondre aux pics de demande ou aux besoins d'expansion. Des problèmes de performance et de disponibilité du système peuvent en résulter.

d. Architecture

En raison de l'architecture monolithique de SAP CX, il est pratiquement impossible d'utiliser une architecture basée sur des micro-services, ce qui prive le client des avantages d'une telle approche modulaire.

4. Solution envisagée

La solution consiste à créer une abstraction technique des opérations de commerce électronique en adoptant une architecture basée sur des microservices, chacun couvrant un domaine fonctionnel (Cart, Customer, Checkout...) et chacun utilisant des technologies open-source autour de Java EE et Spring. Cela permettrait d'atteindre des objectifs multiples tels que l'amélioration de la performance ou la séparation des responsabilités.

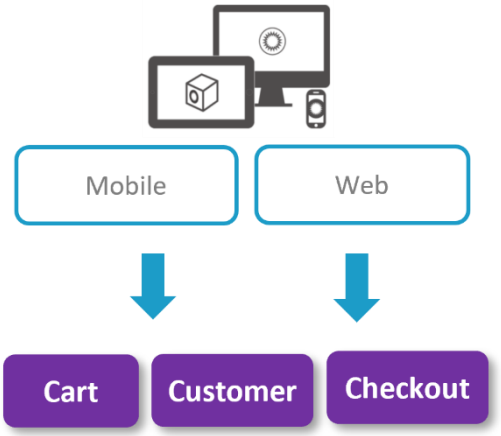


Figure 15: architecture to-be de la plateforme e-commerce NC2

Cependant, la migration vers une telle architecture est délicate, d'où l'idée de mettre en place une architecture transitionnelle via le projet « **eCAPI** ».

Dans le cadre du projet Nespresso, eCAPI (eCommerce API) est une bibliothèque exposant toutes les fonctionnalités e-commerce aux applications clientes (web et natives) sous la forme de services web RESTful.

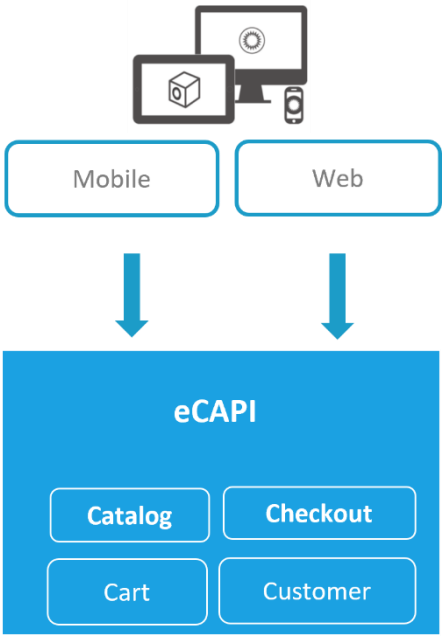


Figure 16: architecture as-is / transitionnelle de la plateforme e-commerce NC2

La transition des services web actuels basés sur Java (eCAPI) vers l'architecture cible (eCAPI Split) sera progressive et spécifique au domaine (par exemple, service web de panier d'achat => micro-service de panier d'achat).

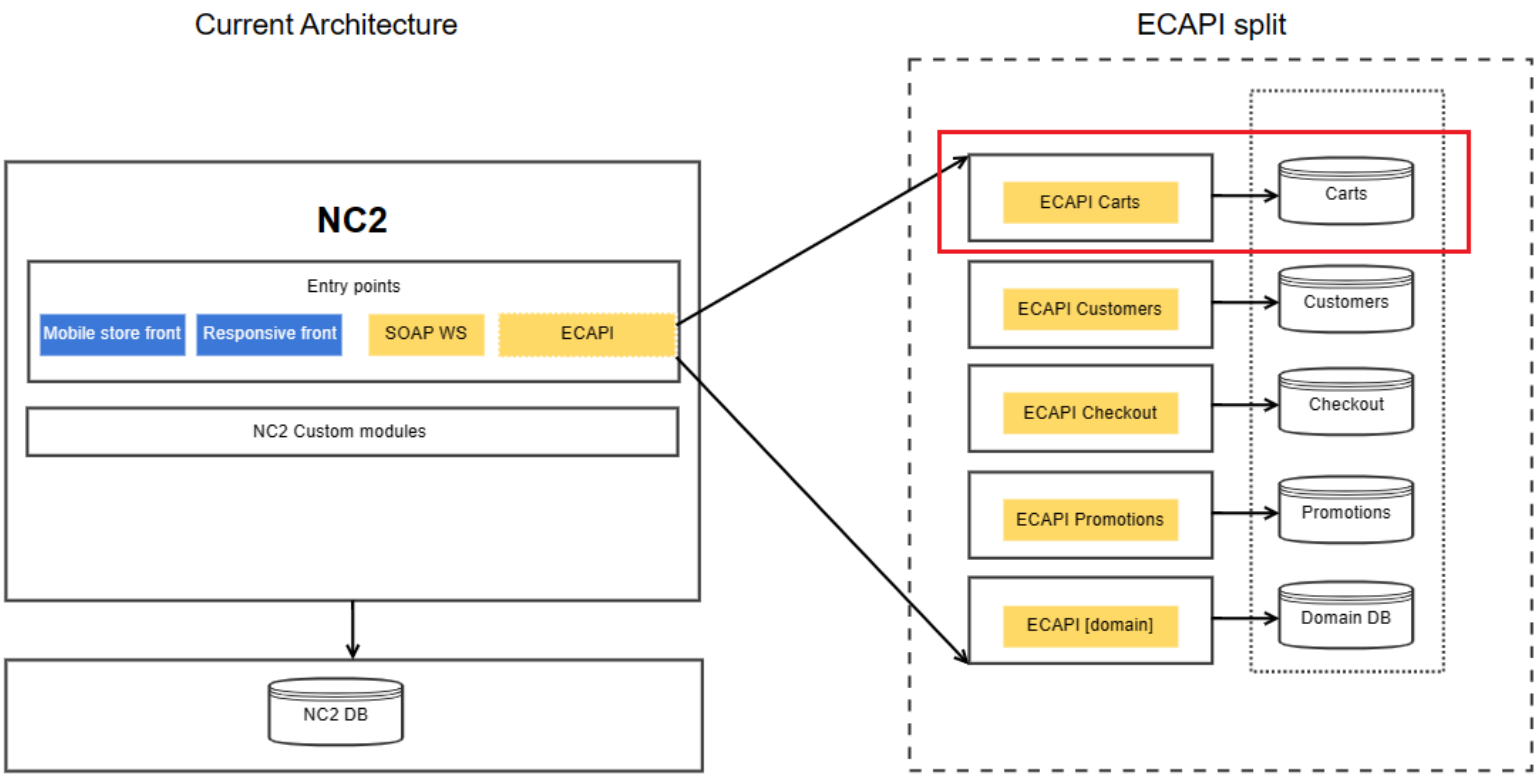


Figure 17: architecture to-be de la plateforme e-commerce NC2

Le reste de ce document aborde en profondeur l'extraction du module Carts vers un répertoire séparé en tant qu'application indépendante (capable d'être déployé en dehors de NC2) qui doit prendre en charge toutes les fonctionnalités ecapi-carts existantes et se conformer au contrat d'API préalablement mis en place.

II. Spécification des exigences

1. Identification des acteurs

Un acteur représente l'abstraction d'un rôle joué par des entités externes qui interagissent directement avec le système étudié. Ainsi, les acteurs susceptibles d'interagir directement avec le module ecapi-carts se présentent comme suit :

Acteur	**Type**	**Description**
eCAPI Client	Primaire	Toute application web ou native utilisant les services exposés par le module ecapi-carts au nom d'un client de Nespresso. On distingue entre un client anonyme et authentifié.
Admin	Secondaire	Ingénieur de support ou développeur disposant de droits d'accès spécifiques.
Scheduler	Secondaire	Agent virtuel qui synchronise périodiquement les données provenant de NC2.

Tableau 2: acteurs et leurs rôles dans le système

2. Identification des exigences fonctionnelles

Le format User Story est devenu la manière la plus populaire d'exprimer les exigences fonctionnelles dans les méthodologies agiles pour un certain nombre de raisons :

- Il se concentre sur le point de vue d'un rôle qui utilisera ou sera impacté par la solution ;
- Il définit l'exigence dans un langage qui a un sens pour ce rôle ;
- Il aide à clarifier la véritable raison de l'exigence ;
- Il permet de définir des exigences de haut niveau sans nécessairement entrer trop tôt dans les détails de bas niveau.

Ci-après sont présentées les fonctionnalités que le module ecapi-carts doit fournir, exprimées sous la forme de User Stories :

- En tant que **client eCAPI**, je dois pouvoir établir un devis pour le panier d'achat non persisté d'un client Nespresso.
- En tant que **client eCAPI authentifié**, je dois pouvoir bénéficier des remises, des cadeaux, et des offres personnalisés au nom d'un client Nespresso.
- En tant que **client eCAPI authentifié**, je dois pouvoir sauvegarder le panier d'achat au nom d'un client Nespresso.
- En tant que **client eCAPI authentifié**, je dois pouvoir fusionner le panier d'achat rempli en mode anonyme avec le panier d'achat préalablement persisté lors de l'authentification d'un client Nespresso.
- En tant que **client eCAPI authentifié**, je dois pouvoir mettre à jour le panier d'achat persisté d'un client Nespresso.
- En tant que **client eCAPI authentifié**, je dois pouvoir supprimer le panier d'achat persisté d'un client Nespresso une fois que le processus de paiement est terminé.
- En tant que **Scheduler**, je dois pouvoir planifier des jobs de synchronisation des données.
- En tant que **Admin**, je dois pouvoir exécuter immédiatement des jobs de synchronisation des données.
- En tant que **Admin**, je dois pouvoir récupérer la configuration courante de l'application.

3. Identification des exigences techniques

Le module ecapi-carts est conçu de telle sorte qu'il répond aux critères suivants :

a. Rapidité

En optant pour une architecture légère et efficace qui met en œuvre des mécanismes de mise en cache (stockage de données en mémoire) et d'équilibrage de charge.

L'un des objectifs à long terme du projet Nespresso est de faire en sorte que le temps de construction et de déploiement ne dépasse pas quelques minutes, contre environ 2 heures actuellement.

b. Scalabilité

En encourageant l'utilisation des techniques de conteneurisation et d'orchestration. Les conteneurs constituent des environnements légers et isolés pour l'exécution des microservices, assurant ainsi la cohérence et la portabilité.

Les plateformes d'orchestration de conteneurs automatisent le déploiement, la mise à l'échelle et la gestion des microservices. Par exemple, elles peuvent adapter dynamiquement le nombre de conteneurs exécutant un microservice spécifique en fonction de certains paramètres prédéfinis tels que la consommation du processeur ou de la mémoire.

c. Résilience maximale et tolérance aux pannes

Atteindre une résilience maximale dans l'architecture cible nécessite de suivre une approche holistique englobant :

- **Une tolérance aux pannes** : à travers la mise en œuvre des mécanismes tels que les patrons Circuit Breaker, Retry et Timeout ;
- **Une bonne gestion des erreurs** : facilitant ainsi l'identification des bugs, et permettant aux développeurs de diagnostiquer et de résoudre les problèmes plus efficacement ;
- **Un monitoring exhaustif** : cela comprend la collecte et l'analyse en temps réel de données sur l'état de santé du système, les mesures de performance et les taux d'erreur.
- **Une récupération totale des données en cas de sinistre** : cela implique de sauvegarder régulièrement les données critiques et de mettre en place des stratégies de réplication des données.

III. Conception

1. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation ci-dessous présente graphiquement les fonctionnalités du module ecapi-carts, en illustrant ses interactions avec les acteurs concernés.

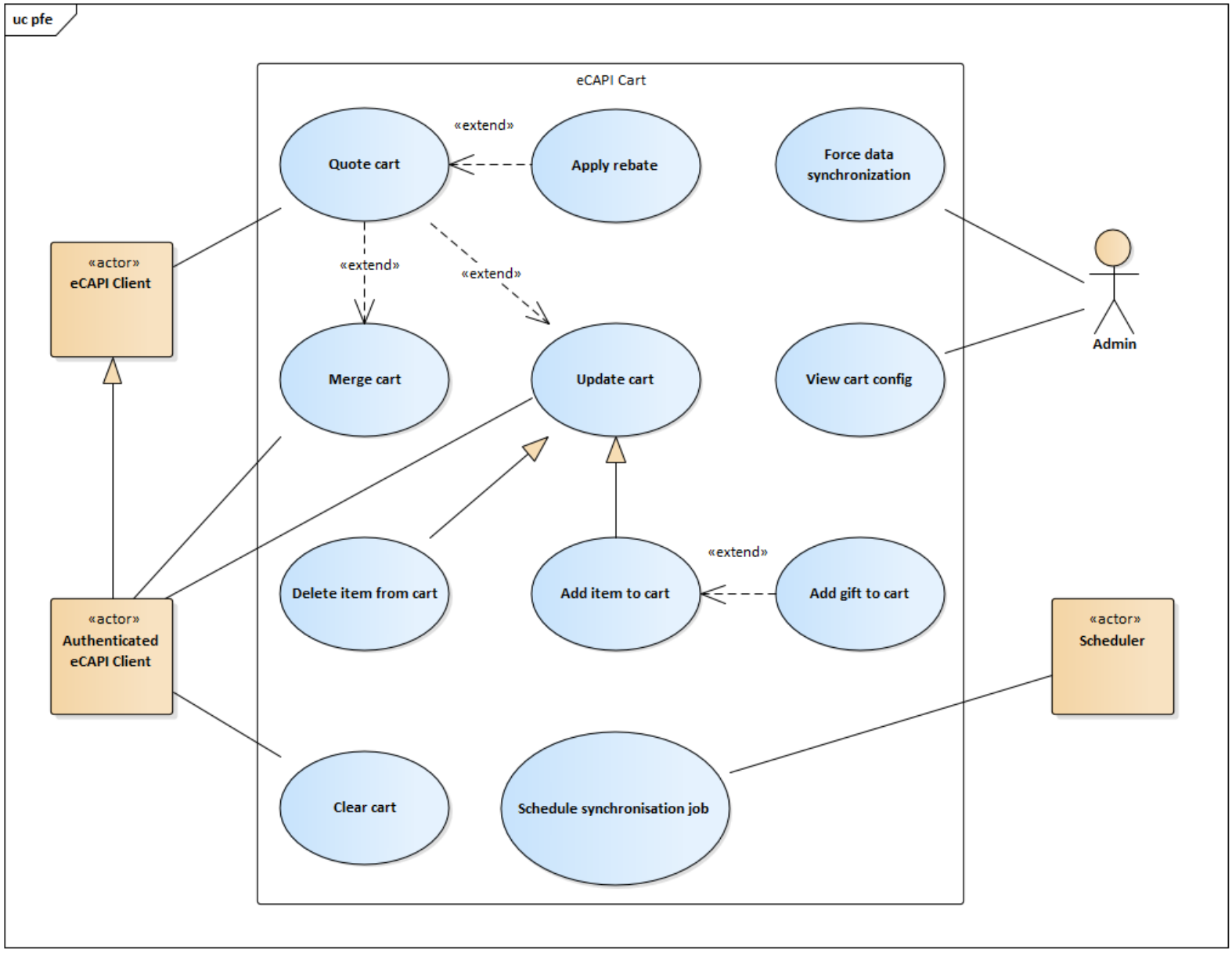


Figure 18: diagramme de cas d'utilisation du module eCAPI Cart

2. Conception de l'API

Le tableau suivant énumère les différentes ressources que le module ecapi-carts doit exposer aux différents acteurs pour accès externe.

Les paramètres "market" et "channel" représentent respectivement le pays de vente (France, Suisse, Royaume-Uni ...) et le canal de vente (B2C ou B2B).

Méthode	Ressource	Description
POST	/{market}/{channel}/anonymous/quoted	Etablit un devis pour le panier d'achat d'un client eCAPI anonyme.
POST	/{market}/{channel}/me/quoted	Fusionne le panier d'achat et établit un devis pour un client eCAPI authentifié.
PUT	/{market}/{channel}/me/quoted	Met à jour le panier d'achat persisté et établit un devis pour un client eCAPI authentifié.
POST	/{market}/{channel}/me/unquoted	Fusionne le panier d'achat sans établir un devis pour un client eCAPI authentifié.
PUT	/{market}/{channel}/me/unquoted	Met à jour le panier d'achat persisté sans établir un devis pour un client eCAPI authentifié.
DELETE	/{market}/{channel}/me	Supprime le panier d'achat persisté d'un client eCAPI authentifié.
GET	/{market}/{channel}/config	Retourne la configuration courante de l'application.

Tableau 3: conception de l'API

3. Diagrammes de séquence

Cette partie fait appel aux diagrammes de séquence afin de représenter le comportement dynamique souhaité du module ecapi-carts lors de l'exécution des divers scénarios.

a. Cas d'utilisation : établir un devis d'achat pour un client anonyme

Chaque fois qu'un client anonyme ajoute ou retire des articles de son panier, celui-ci est mis à jour, mais uniquement dans la session en cours.

Si l'option de cotation est activée pour le marché/canal correspondant, un panier avec devis est généré après chaque modification, fournissant au client les prix exacts des articles sélectionnés.

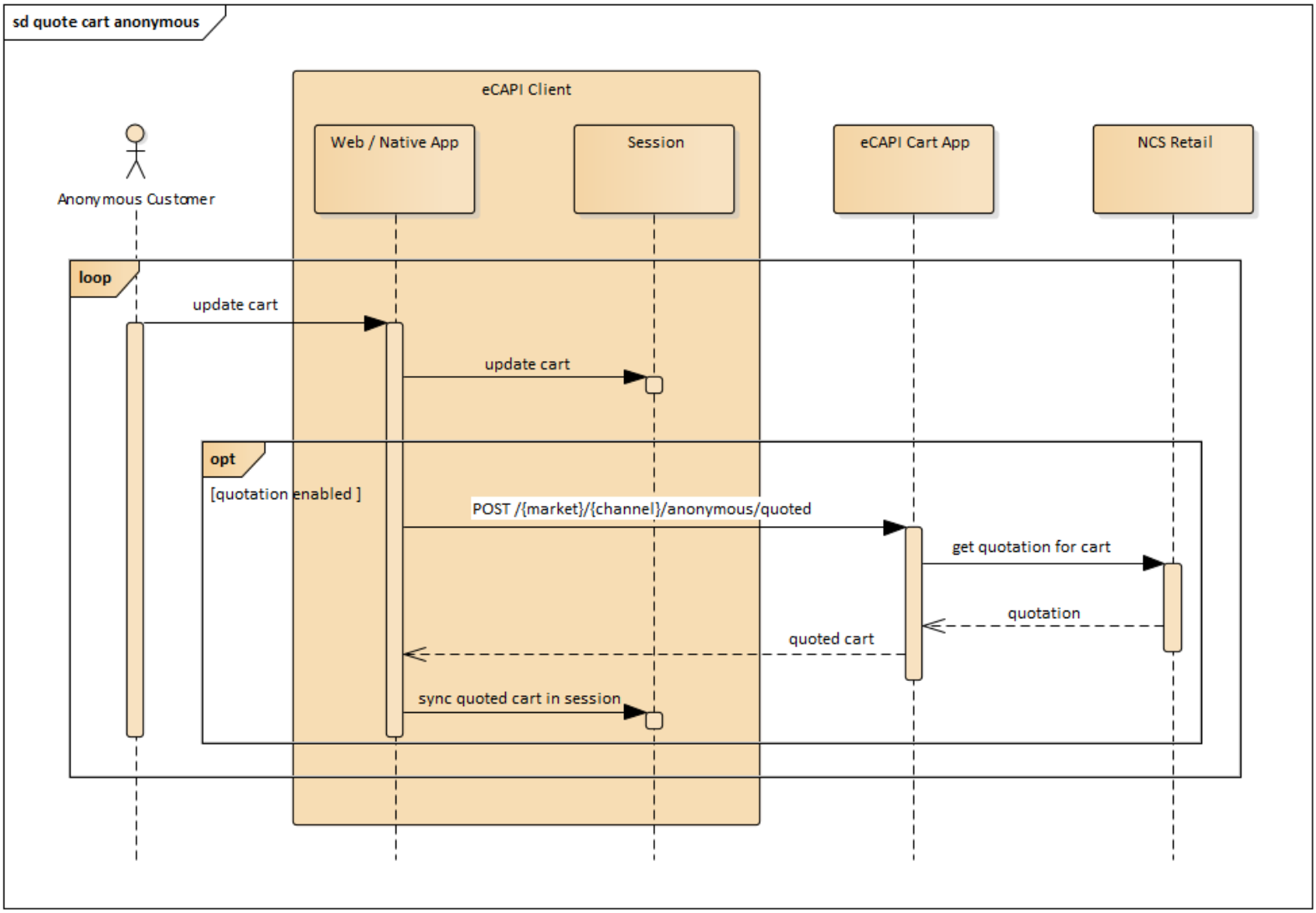


Figure 19: diagramme de séquence « établir un devis d'achat pour un client anonyme »

b. Cas d'utilisation : fusionner le panier d'achat lors de l'authentification

Une fois qu'un client anonyme est authentifié, le panier d'achat dans sa session est fusionné avec le panier d'achat précédemment persisté dans son compte.

Si l'option de cotation est activée pour le marché/canal correspondant, un panier avec devis est généré et le panier de la session est mis à jour.

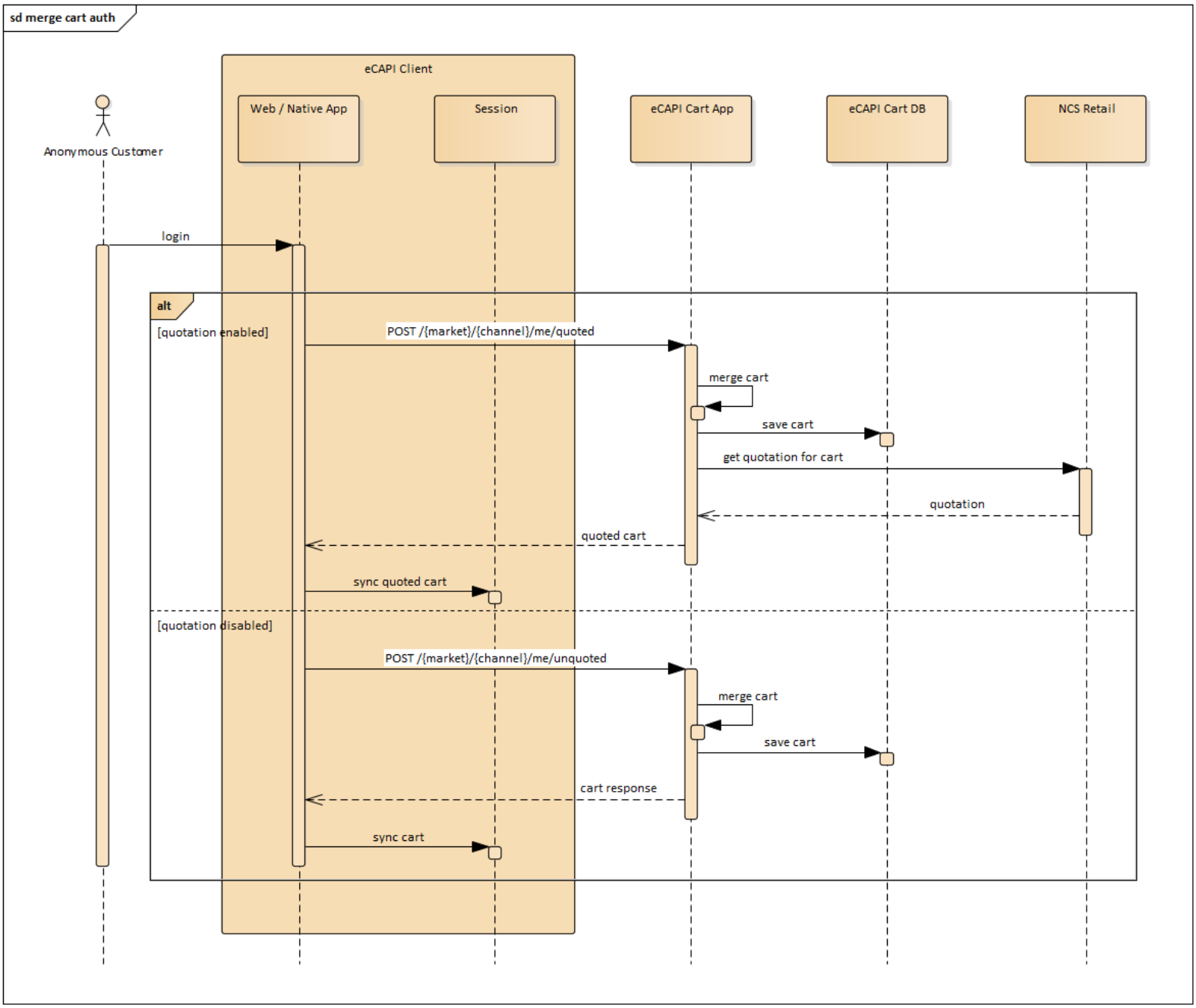


Figure 20 : diagramme de séquence « fusionner le panier d'achat lors de l'authentification »

c. Cas d'utilisation technique : synchroniser les données depuis NC2

Le Scheduler planifie des jobs de synchronisation de données à exécuter par l'application web, chaque job récupère les données depuis NC2 puis met à jour la base de données du module.

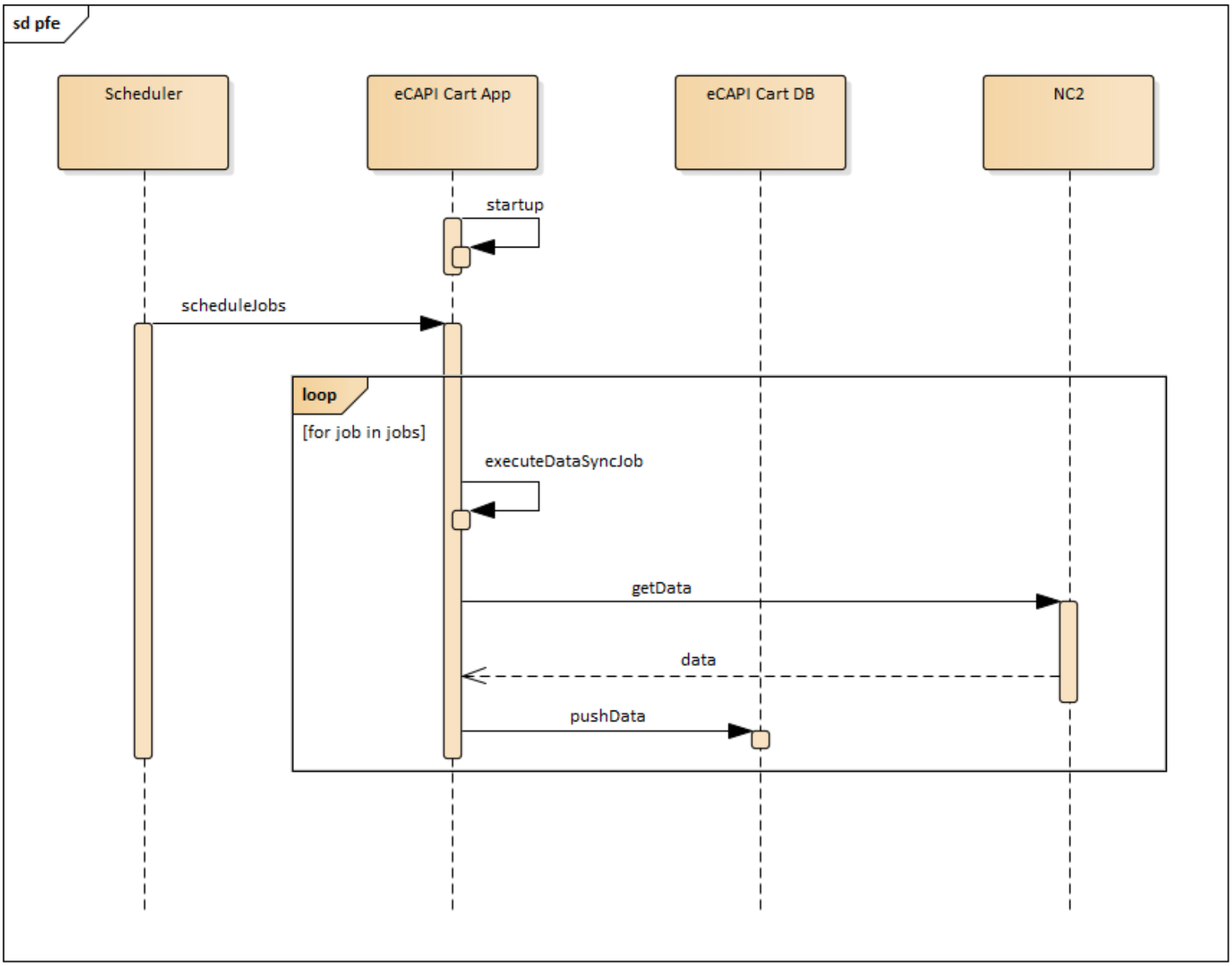


Figure 21: diagramme de séquence « synchroniser les données depuis NC2 »

d. Cas d'utilisation technique : mettre en cache les données

Dans le module ecapi-carts, les données relatives aux produits sont mises en cache, si elles ne sont pas disponibles dans le cache, elles sont récupérées à partir de la base de données et le cache est mis à jour.

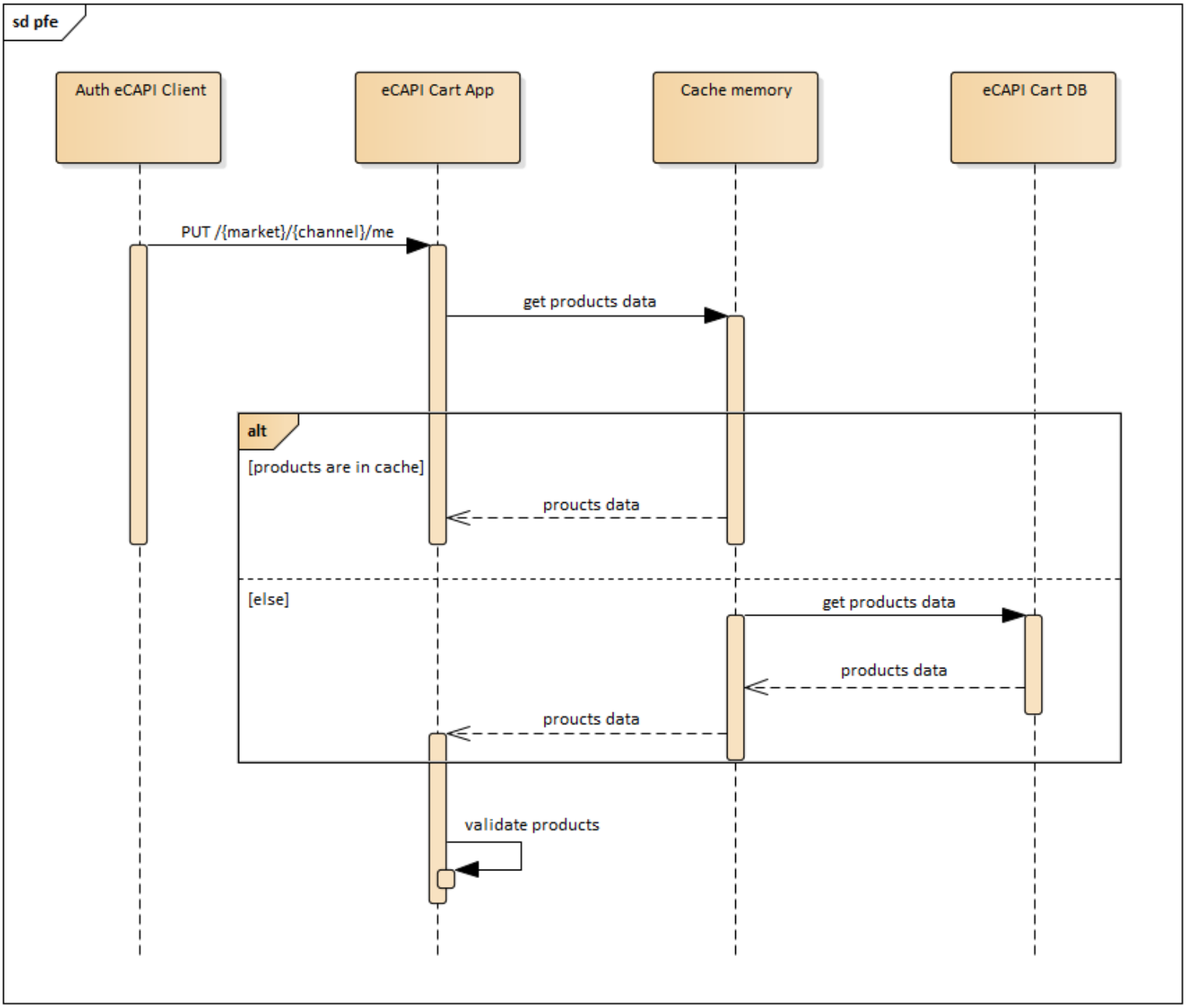


Figure 22: diagramme de séquence « mettre en cache les données »

Une analyse approfondie de la solution existante était indispensable pour saisir avec précision les besoins fonctionnels et techniques de la solution cible. Cette analyse a également permis de créer un schéma conceptuel pour la mise en œuvre de la nouvelle solution.

Le choix des outils et des technologies à utiliser sera basé sur cette analyse, ainsi que sur d'autres facteurs. Le chapitre suivant abordera cette étape plus en détail.

Environnement et outils

Chapitre 3 : Environnement et outils

Ce chapitre présente les outils choisis pour développer le module ecapi-carts et/ou la totalité du projet eCAPI Split, comment ils sont utilisés et pourquoi ils ont été choisis.

I. Développement

1. Langage : Java

Le langage de programmation Java est un langage à usage général, concurrent, fortement typé, orienté objet et basé sur la notion de classes. [6]

Java est un choix populaire pour les applications d'entreprise en raison de son évolutivité, de sa compatibilité multiplateforme, de sa sécurité, de son vaste écosystème et de l'appui solide de la communauté.

Figure 23 : logo Java

2. Framework : Spring / Spring Boot

Spring est un cadre de développement open source pour construire et définir l'infrastructure d'une application Java, dont il facilite le développement et les tests.

Spring s'appuie principalement sur l'intégration de trois concepts clés :

- L'inversion de contrôle est assurée de deux façons différentes : la recherche de dépendances et l'injection de dépendances ;
- La programmation orientée aspect ;
- Une couche d'abstraction.

La couche d'abstraction permet d'intégrer d'autres Framework et bibliothèques avec une plus grande facilité. Cela se fait par l'apport ou non de couches d'abstraction spécifiques à des frameworks particuliers. [7]

Figure 24: logo Spring

Spring Boot est un cadre de développement d'applications en Java qui utilise Spring comme couche de base pour construire une application autonome avec une configuration inexistante ou minimale. Il est spécifiquement utilisé pour la création des APIs REST et des micro-services. [8]

Figure 25: logo Spring Boot

II. Tests

1. JUnit

JUnit est un framework open source pour le développement et l'exécution de tests unitaires automatisables. Le principal intérêt est de s'assurer que le code répond toujours aux besoins même après d'éventuelles modifications. Plus généralement, ce type de tests est appelé tests unitaires de non-régression. [9]

JUnit :

- Prend en charge le développement piloté par les tests (TDD) : une pratique de développement logiciel agile qui consiste à écrire des tests avant d'écrire du code ;
- Peut être intégré à des outils de construction tels que Maven qui permet d'automatiser le processus de test dans le cadre du processus de Build ;
- Prend en charge le mocking, ce qui permet de simuler des dépendances complexes et de tester le code de manière isolée.

Figure 26: logo JUnit

2. Mockito

Mockito est un framework Java très connu permettant de générer automatiquement des objets simulés. Couplé avec JUnit, il permet de tester le comportement des objets réels associés à un ou des objets simulés, facilitant ainsi l'écriture des tests unitaires. [10]

Mockito a plusieurs usages dans le module ecapi-carts :

- **Simulation des dépendances** : Mockito permet de créer des objets fictifs qui simulent des dépendances dans le système testé ;
- **Simplification des tests unitaires** : Mockito simplifie le processus d'écriture des tests unitaires en fournissant une API simple et intuitive pour la création d'objets simulés ;
- **Vérification du comportement** : Mockito fournit une gamme de méthodes pour vérifier le comportement des objets simulés, comme la vérification qu'une méthode a été appelée avec des arguments spécifiques ou qu'une méthode a été appelée un certain nombre de fois.

Figure 27: logo Mockito

III. Construction de projet

1. Apache Maven

Maven est un outil de construction de projets (build) open-source développé par la fondation Apache. Il est utilisé dans le projet eCAPI Split pour :

- Gérer les dépendances ;
- Structurer le projet ;
- Automatiser la compilation et l'exécution des tests unitaires ;
- Générer des documentations concernant le projet.

Figure 28: logo Maven

IV. SGBD

1. Apache Cassandra

Apache Cassandra est un système de gestion de base de données (SGBD) de type NoSQL conçu pour gérer des quantités massives de données sur un grand nombre de serveurs, assurant une haute disponibilité en éliminant les points de défaillance unique. Il permet une répartition robuste sur plusieurs centres de données, avec une réplication asynchrone sans nœud maître et une faible latence pour les opérations de tous les clients. Le projet est publié en logiciel libre et porté par la fondation Apache. [11]

Figure 29: logo Apache Cassandra

Cassandra se révèle être l'option optimale pour la persistance des données dans le module ecapi-cart, et ce pour diverses raisons :

- **Schéma flexible** : les données du module ecapi-cart sont en constante mutation et ne requièrent pas de schéma.

- **Évolutivité horizontale** : Cassandra permet d'ajouter du matériel pour accueillir plus de clients et plus de données, selon les besoins.
- **Architecture AlwaysON** : Cassandra n'a pas de point de défaillance unique et est disponible en permanence pour les applications critiques qui ne peuvent pas se permettre une défaillance.
- **Prise en charge des transactions** : Cassandra prend en charge les propriétés ACID (l'atomicité, la cohérence, l'isolation et la durabilité).
- **Écritures rapides** : Cassandra effectue des écritures extrêmement rapides et peut stocker des centaines de téraoctets de données, sans sacrifier la rapidité de la lecture.

V. IDEs

1. Intellij IDEA

IntelliJ IDEA est un environnement de développement intégré, destiné au développement de logiciels informatiques reposant sur la technologie Java. Il est développé par JetBrains (anciennement « IntelliJ ») et disponible en deux versions, l'une communautaire, open source, sous licence Apache 2 et l'autre propriétaire, protégée par une licence commerciale. Tous deux supportent les langages de programmation Java, Kotlin, Groovy et Scala. [12]

Figure 30: logo Intellij IDEA

VI. Gestion de code source

1. Git

Git est un système de contrôle de version distribué qui permet de suivre les modifications apportées à un ensemble de fichiers. Il est généralement utilisé pour coordonner le travail des programmeurs travaillant en collaboration sur le code source lors du développement d'un logiciel. Ses avantages sont la rapidité, l'intégrité des données et la prise en charge de flux de travail distribués et non linéaires. [13]

Figure 31: logo Git

2. Atlassian Bitbucket

Bitbucket est une combinaison de serveur Git et d'interface web écrite en Java et construite avec Apache Maven. Il permet aux utilisateurs d'effectuer des opérations Git de base (comme la révision ou la fusion de code) tout en contrôlant l'accès en lecture et en écriture au code. Il assure également l'intégration avec d'autres outils d'Atlassian. [14]

Figure 32: logo Bitbucket

Bitbucket était un choix évident pour héberger le code source du projet eCAPI Split, et ce pour diverses raisons :

- **Intégration avec les outils d'Atlassian** : Bitbucket appartient à Atlassian, qui possède également Jira, Confluence et d'autres outils de gestion de projet populaires ;
- **Meilleure prise en charge de Git** : Bitbucket a été conçu spécifiquement pour Git, contrairement à d'autres serveurs ;
- **Meilleure tarification pour les répertoires privés** : Bitbucket offre des répertoires privés gratuits pour un maximum de 5 utilisateurs, et propose des plans tarifaires plus abordables pour les équipes plus importantes par rapport à d'autres options ;
- **Meilleur contrôle des permissions** : Bitbucket offre un contrôle plus granulaire des permissions pour les répertoires, les branches et les fichiers ;
- **Meilleure sécurité** : Bitbucket offre des fonctionnalités de sécurité de niveau entreprise telles que l'IP whitelisting, l'authentification à deux facteurs et l'authentification unique SAML ;
- **Pipelines personnalisables** : Bitbucket fournit des pipelines paramétrables qui peuvent être utilisés pour automatiser le build, les tests et les déploiements de logiciels.

VII. DevOps

1. Docker

Docker est une plate-forme logicielle permettant de concevoir, tester et déployer des applications rapidement. Docker intègre les logiciels dans des unités normalisées appelées conteneurs, qui rassemblent tous les éléments nécessaires à leur fonctionnement, dont les bibliothèques, les outils système, le code et l'environnement d'exécution. [15]

Figure 33: logo Docker

2. Jenkins

Jenkins est un outil d'automatisation open-source écrit en Java avec des plugins conçus pour l'intégration continue. Jenkins est utilisé pour construire et tester des projets logiciels en continu, ce qui permet aux développeurs d'intégrer plus facilement les modifications apportées au projet et aux utilisateurs d'obtenir plus facilement une nouvelle version. Il permet également la livraison continue de logiciels en s'intégrant à un large éventail de technologies de test et de déploiement. [16]

Figure 34: logo Jenkins

Jenkins accélère le processus de développement logiciel d'eCAPI Split grâce à l'automatisation de tous les processus de son cycle de vie, y compris le build, la documentation, les tests, le packaging, le staging, le déploiement et l'analyse statique du code source.

3. Kubernetes

Kubernetes, également connu sous le nom de K8s, est un système open-source permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Il regroupe les conteneurs qui composent une application en unités logiques pour en faciliter la gestion et la découverte. [17]

Figure 35: logo Kubernetes

Dans le cadre du projet eCAPI Split, Kubernetes est utilisé pour :

- Le déploiement et mise à l'échelle automatisés des instances à la demande ;
- La haute disponibilité et la tolérance aux pannes : réplication automatique des microservices et prise en charge de l'auto-réparation ;
- La découverte des microservices et l'équilibrage de la charge : attribution des adresses IP aux services et répartition du trafic entre plusieurs instances d'un service ;
- La surveillance et la gestion : utilisation des outils de surveillance natifs de Kubernetes tels que Prometheus et Grafana, ou des solutions tierces, pour surveiller l'état de santé et l'utilisation des ressources des différents modules.

VIII. Développement d'APIs

1. Swagger

Swagger est un langage de description d'interface permettant de décrire des API exprimées à l'aide de JSON. Swagger est utilisé avec toute une série d'outils logiciels open source pour concevoir, créer, documenter et utiliser des services Web. [18]

Figure 36: logo Swagger

Dans le cadre du projet eCAPI Split, Swagger est utilisé pour :

- **La conception d'eCAPI** : Swagger fournit un ensemble d'outils puissants pour la conception d'API, y compris un éditeur visuel et des moyens de génération de code ;
- **La documentation d'eCAPI** : Swagger fournit une documentation automatique pour les APIs, y compris des descriptions de tous les endpoints, paramètres et réponses possibles ;
- **La standardisation** : Swagger fournit un moyen standard de décrire les API ;
- **La collaboration** : Swagger fournit un environnement collaboratif qui permet aux développeurs de travailler ensemble sur la conception et le développement des API.

2. Postman

Postman est un outil de développement d'API populaire utilisé par les développeurs généralement pour tester les APIs. Il fournit une interface graphique conviviale pour effectuer des requêtes d'API et visualiser les réponses dans un format facile à lire. Il prend également en charge des fonctionnalités telles que l'authentification, les tests automatisés et la génération de documentation.

Figure 37: logo Postman

IX. Gestion de projet

1. Atlassian Jira

Jira est une application logicielle qui a été développée par l'entreprise australienne de logiciels Atlassian et qui permet aux équipes de suivre les problèmes, de gérer les projets et d'automatiser les flux de travail.

Figure 38: logo Jira

Jira est basé sur quatre concepts clés :

- **Ticket** : un ticket est un élément de travail unique à suivre depuis sa création jusqu'à son achèvement. Il peut s'agir d'un bug, d'une user story, d'un epic, d'une tâche à accomplir humaines ou d'un artefact que l'équipe de documentation doit créer ;
- **Projet** : un projet est un moyen de regrouper les tickets ainsi que les informations et le contexte communs qui les relient ;
- **Tableau** : un tableau dans Jira est une représentation visuelle du workflow de l'équipe au sein d'un projet ;

- **Workflow** : un workflow représente le chemin que prennent les tickets au fur et à mesure qu'ils progressent dans le projet, de la création à l'achèvement. Chaque étiquette d'un workflow, telle que *To Do*, *In Progress*, et *Done*, représente un statut qu'un ticket peut avoir. [19]

2. Atlassian Confluence

Confluence est un logiciel populaire développé par Atlassian qui permet aux membres du personnel de conserver toutes les informations relatives à un projet particulier en un seul endroit, ce qui leur permet d'accéder aux rapports, aux notes de réunion, aux brouillons, aux listes de tâches et à bien d'autres choses encore.

Figure 39: logo Confluence

Certaines caractéristiques utiles de Confluence sont les suivantes :

- **Création une documentation consultable** ;
- **Gestion efficace des tâches** : Confluence offre une mise en œuvre efficace des tâches au sein d'une interface unique ;
- **Prise de décision organisée** : Confluence permet aux membres de l'équipe de créer des pages pour discuter et faire progresser les décisions ;
- **Partage de fichiers simple** : Confluence permet aux utilisateurs de partager des fichiers de différentes manières ;
- **Gestion de projet** : Confluence permet une gestion de projet agile et flexible, quelle que soit la façon dont les managers préfèrent travailler. [20]

X. Suivi de performance et révision de code

1. Prometheus

Prometheus est un logiciel libre de surveillance informatique et générateur d'alertes. Il enregistre des métriques en temps réel dans une base de données de séries temporelles (avec une capacité d'acquisition élevée) en se basant sur le contenu de point d'entrée exposé à l'aide du protocole HTTP. [21]

Figure 40: logo Prometheus

2. Grafana

Grafana est une application web open source multiplateforme d'analyse et de visualisation interactive. Elle fournit des diagrammes, des graphiques et des alertes pour le web lorsqu'elle est connectée à des sources de données prises en charge. [22]

Figure 41: logo Grafana

Les deux solutions sont essentiellement complémentaires, malgré quelques chevauchements. Prometheus collecte des métriques riches et fournit un langage d'interrogation puissant tandis que Grafana transforme ses métriques en visualisations significatives.

3. SonarQube

SonarQube est un logiciel libre de qualimétrie en continu de code. Il aide à la détection, la classification et la résolution de défaut dans le code source, permet d'identifier les duplications de code, de mesurer le niveau de documentation et de connaître la couverture de test déployée.

Figure 42: logo SonarQube

SonarQube permet une surveillance continue de la qualité du code grâce à son interface web permettant de voir les défauts de l'ensemble du code et ceux ajoutés par la nouvelle version. Le logiciel peut être interfacé avec un système d'automatisation comme Jenkins pour inclure l'analyse comme une extension du développement. [23]

La solution Sonar effectue des vérifications à chaque étape du processus de développement :

- SonarLint fournit un retour d'information immédiat dans l'IDE pendant que les développeurs écrivent leur code, ce qui leur permet de trouver et de corriger les problèmes avant de valider le code.
- Les barrières de qualité empêchent le code présentant des problèmes d'être mis en production, ce qui est un outil clé pour aider les développeurs à intégrer la méthodologie « Clean as You Code » permettant aux développeurs de se concentrer sur la soumission d'un nouveau code propre pour la production, tout en sachant que le code existant sera amélioré au fil du temps. [24]

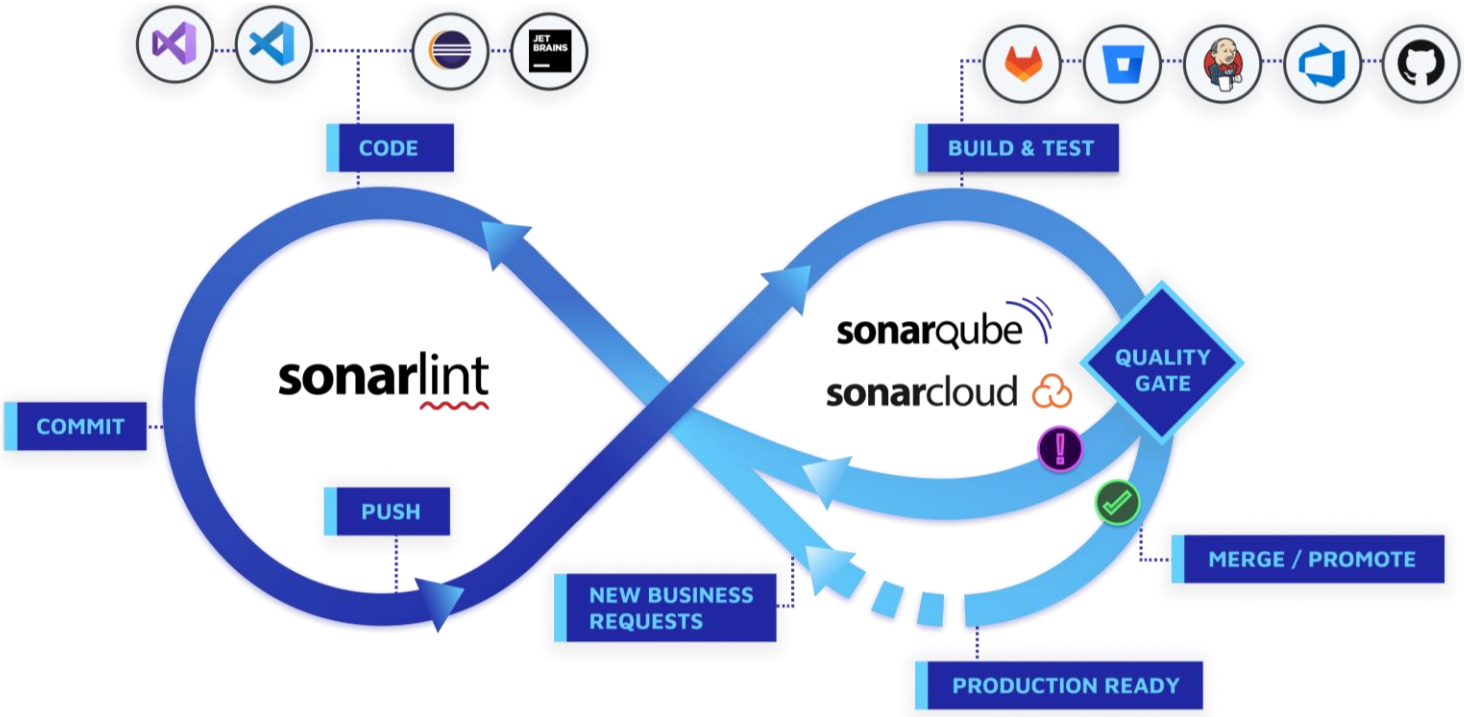


Figure 43: la solution Sonar

Ce chapitre a souligné les outils et technologies spécifiques qui seront utilisés pour développer le projet eCAPI Split. Ces outils et technologies ont été sélectionnés en fonction de leur pertinence, de leur fiabilité et de leur compatibilité avec les exigences du projet. Le chapitre suivant se focalisera sur la concrétisation du module ecapi-carts.

Réalisation

Chapitre 4 : Réalisation

Ce chapitre présente l'architecture du module ecapi-carts et la structure de ses sous-modules, et décrit l'interface qu'il expose, ainsi qu'un scénario d'utilisation de celle-ci du point de vue d'un utilisateur final.

I. Architecture technique du module ecapi-carts

L'architecture technique du module est mise en place dans un cluster Kubernetes sur site tel que :

- Chaque instance ecapi-carts est un pod Kubernetes composé d'un seul conteneur hébergeant l'application web ;
- La base de données est configurée comme un service unique que toutes les instances consomment ;
- Un cache en mémoire est utilisé pour stocker les données fréquemment consultées ;
- Un pod expose trois interfaces de type REST à ses clients (client anonyme, client authentifié et administrateur) et s'appuie sur deux interfaces externes (NC2 et Retail).

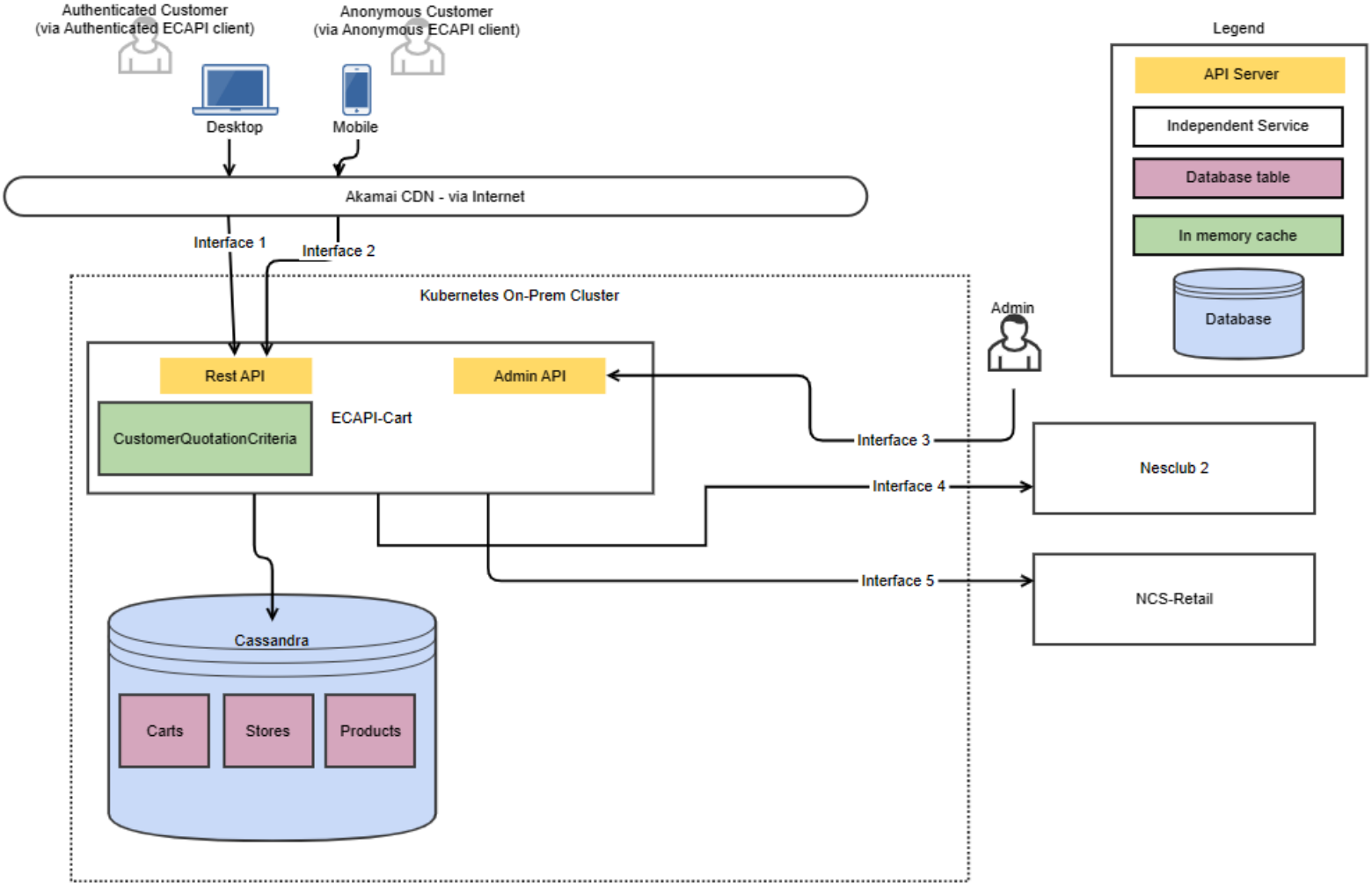


Figure 44: architecture technique du module ecapi-carts

II. Structure du projet

1. Domain Driven Design

Le Domain-Driven Design (DDD) est un ensemble de principes et de patrons qui aident les développeurs à concevoir des systèmes orientés objet élégants. Correctement appliquée, cette approche peut conduire à des abstractions logicielles appelées modèles de domaine. Ces modèles encapsulent une logique métier complexe, comblant ainsi les écarts entre la réalité du métier et le code. [25]

2. Architecture en oignon

L'architecture en oignon est un modèle architectural qui propose que les logiciels soient conçus en couches, chaque couche ayant ses propres préoccupations.

La règle d'or de l'architecture est la suivante :

> *« Aucun élément d'un cercle intérieur ne peut savoir quoi que ce soit sur un élément d'un cercle extérieur. Cela inclut les fonctions, les classes, les variables ou toute autre entité logicielle nommée. »*
>
> *Robert C. Martin [26]*

L'architecture en oignon présente plusieurs avantages, elle :

- Permet de créer une logique métier qui n'est liée à rien et par conséquent, le système peut être testé rapidement puisque le cœur de l'application est indépendant ;
- Convient parfaitement aux microservices, où ce n'est pas seulement une base de données qui peut servir de couche d'accès aux données, mais aussi, par exemple, un client http, si le microservice a besoin d'obtenir des données d'un autre microservice, voire d'un système externe.

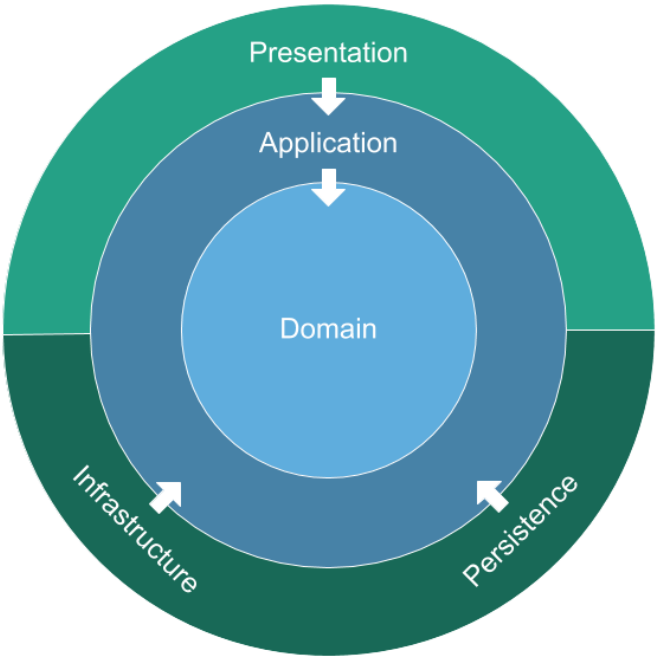


Figure 45: architecture en oignon

L'architecture en oignon se compose des couches suivantes, chacune ayant ses propres responsabilités et dépendances :

- **Couche domaine** : contient les entités métier, les règles métier et la logique du domaine qui régissent le comportement et les opérations du système.
- **Couche application** : contient la logique propre à l'application et orchestre les interactions entre les entités du domaine afin de réaliser des cas d'utilisation ou des opérations spécifiques.
- **Couche infrastructure** : fournit les détails de la mise en œuvre technique. Elle comprend les composants responsables de l'accès aux données (persistance), des services externes, de la journalisation, de la mise en cache, etc.
- **Couche de présentation** : responsable du rendu de l'interface utilisateur et de la gestion des interactions avec l'utilisateur. Elle peut être constituée de divers composants tels que des applications web, des applications natives ou des API.

3. Structure multi-modules

Le projet ecapi-carts est structuré comme un projet multi-modules qui adhère aux principes du Domain Driven Design en se basant sur une architecture en oignon.

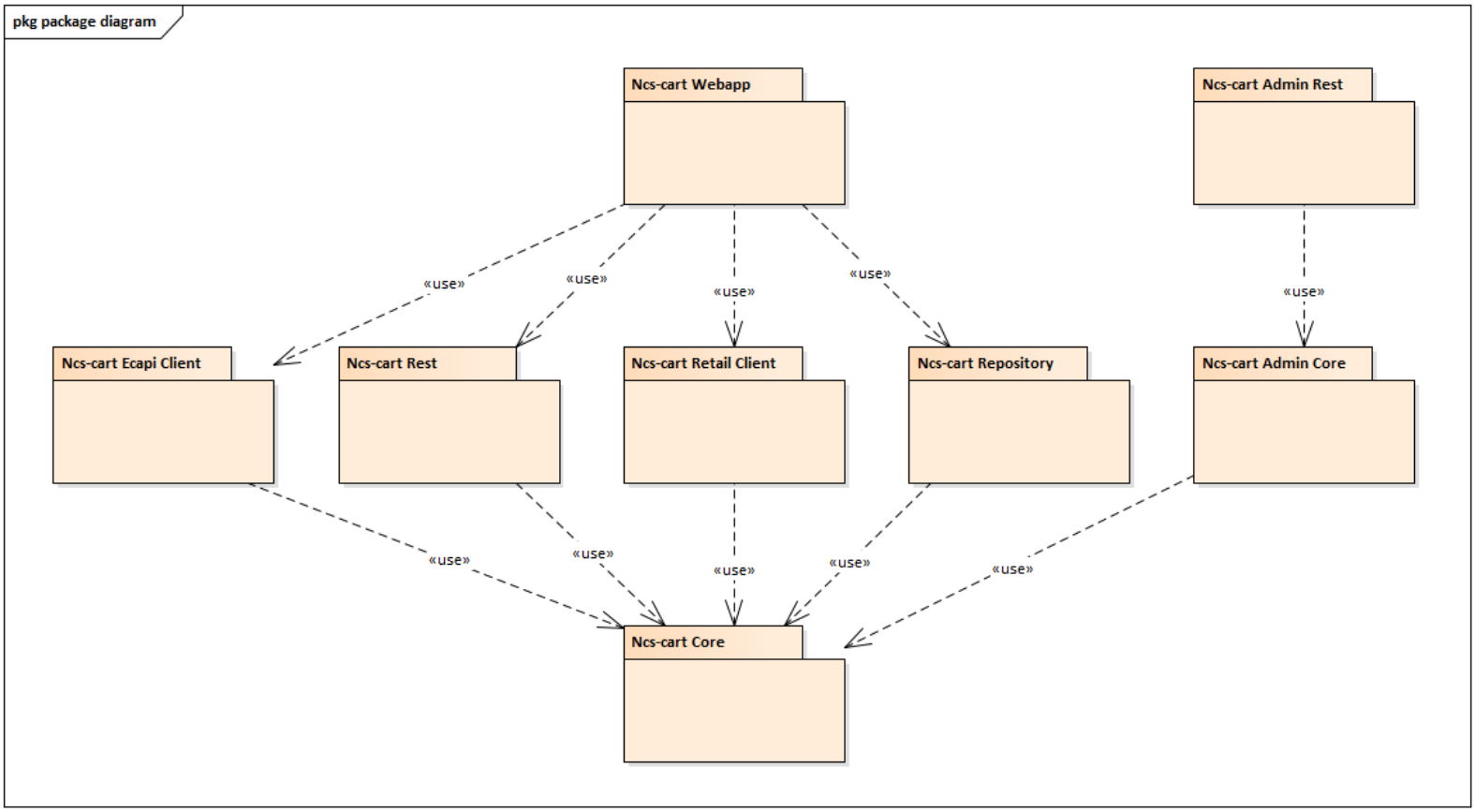


Figure 46: sous modules du projet ecapi-carts

Ci-dessous sont décrits les rôles de chaque module :

- **Ncs-cart Webapp** : module principal qui regroupe tous les éléments des autres modules en un seul artefact ;

- **Ncs-cart Rest** : expose les endpoints ecom-api-carts Rest ;
- **Ncs-cart Ecapi Client** :communique avec d'autres services ecapi pour récupérer les données nécessaires (par exemple, ecapi-customer pour récupérer les données de l'utilisateur authentifié) ;
- **Ncs-cart Repository** : lit et stocke les données de/vers la base de données Cassandra par l'intermédiaire de la bibliothèque de persistance ;
- **Ncs-cart Retail Client** : définit le client Retail permettant d'établir le devis d'un panier ;
- **Ncs-cart Core** : contient la logique commune exploitable par les autres modules, et définit les contrats que les autres modules doivent mettre en œuvre ;
- **Ncs-cart Admin Rest** : expose certains endpoints aux administrateurs leur permettant d'obtenir des informations sur la configuration actuelle et d'effectuer des tâches de synchronisation :
  - Forcer la synchronisation pour un marché et un canal donnés ;
  - Obtenir l'état actuel d'une partition de données « Product » pour un marché et un canal donnés ;
  - Obtenir le statut global pour un marché donné (les deux canaux) ;
  - Obtenir des informations de support (version de l'application, dernier commit, branche git, identifiant de commit ...).

La figure suivante présente une vue de la structure du projet dans Intellij Idea.

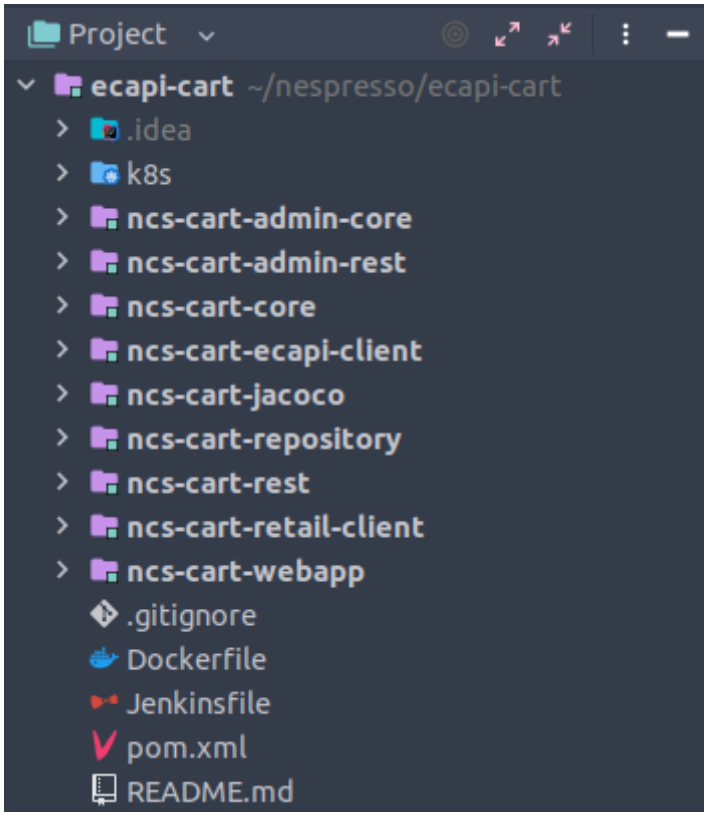


Figure 47: vue de la structure du projet dans Intellij Idea

III. Scénario d'utilisation

Cette section présente un scénario d'utilisation du module ecapi-carts dans une perspective d'utilisateur final b2c possédant un compte Nespresso (l'API est consommée par un client web), le scénario est le suivant :

Le client Nespresso arrive sur la page d'accueil de la plateforme à travers un navigateur web (client eCAPI anonyme).

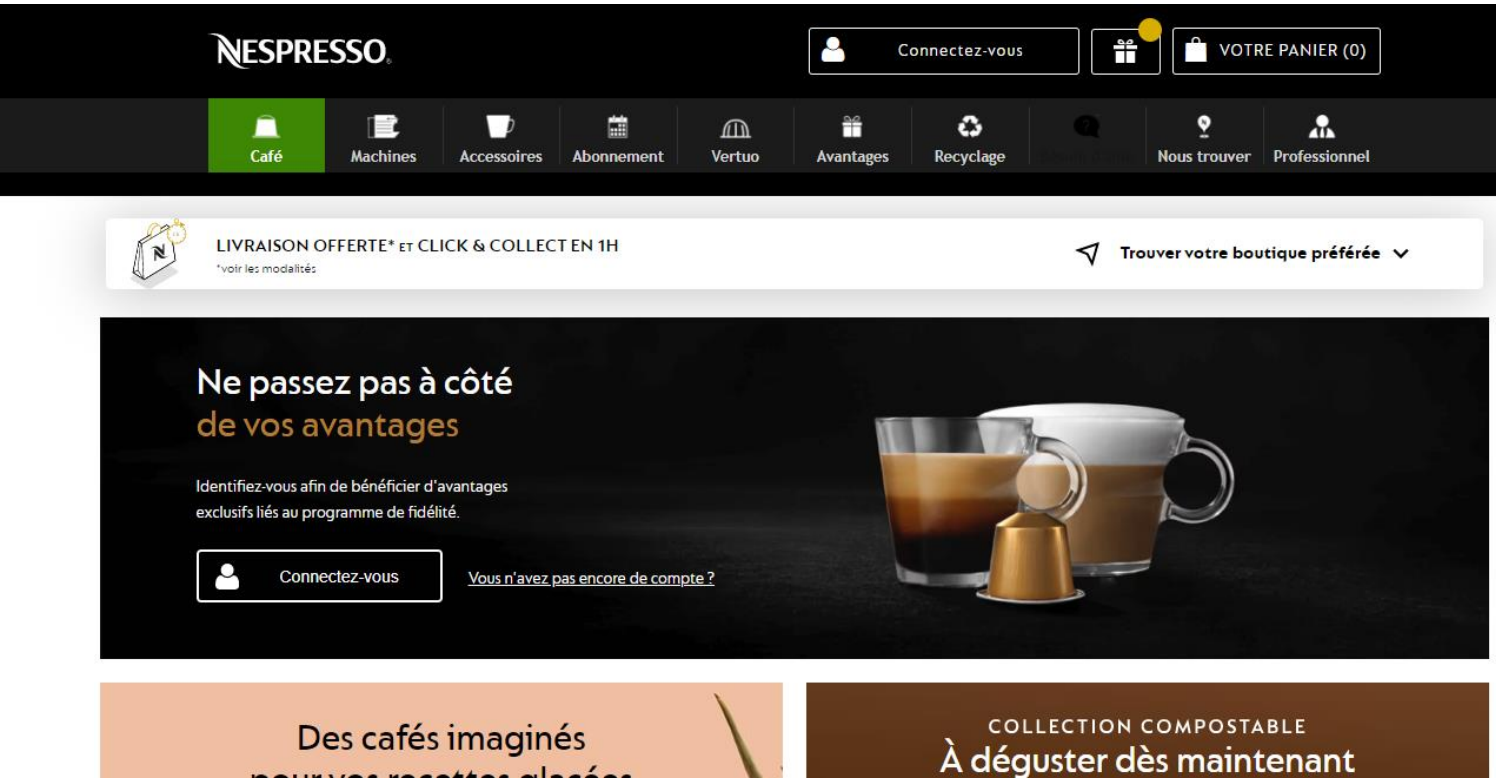


Figure 49: page d'accueil de la plateforme Nespresso

Le panier d'achat d'un client anonyme est initialement vide (évidemment).

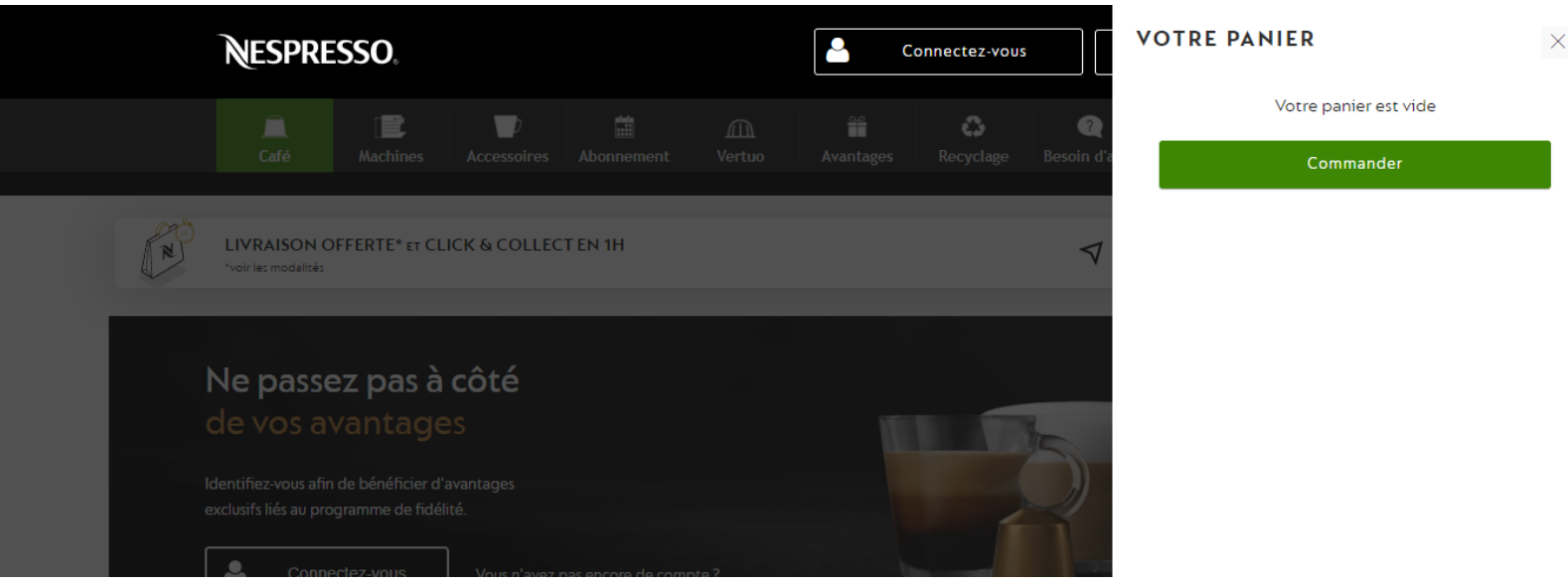


Figure 48: panier d'achat vide

Il parcourt différents produits et catégories, recherchant un article spécifique ou explorant les options.

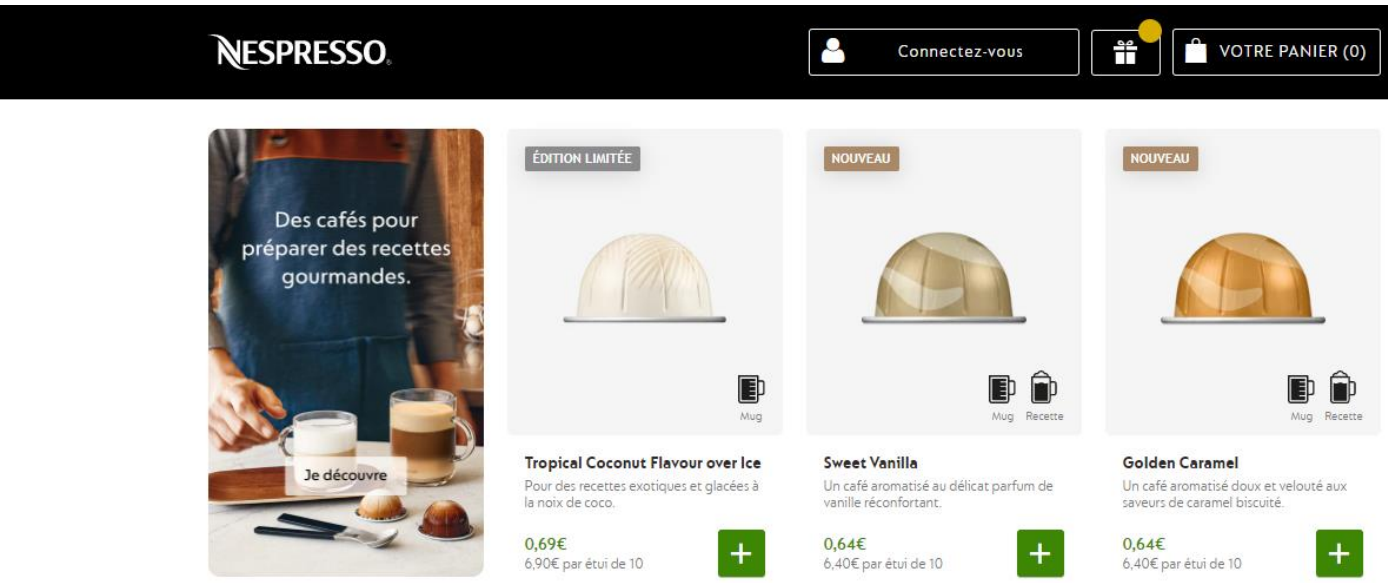


Figure 50: liste des produits

Après avoir trouvé un produit souhaité, le client le sélectionne et consulte ses détails, y compris le prix (non exact) et la description. Satisfait, le client ajoute l'article à son panier d'achat et choisit une quantité.

Jusqu'à présent, le panier n'est mis à jour qu'en session.

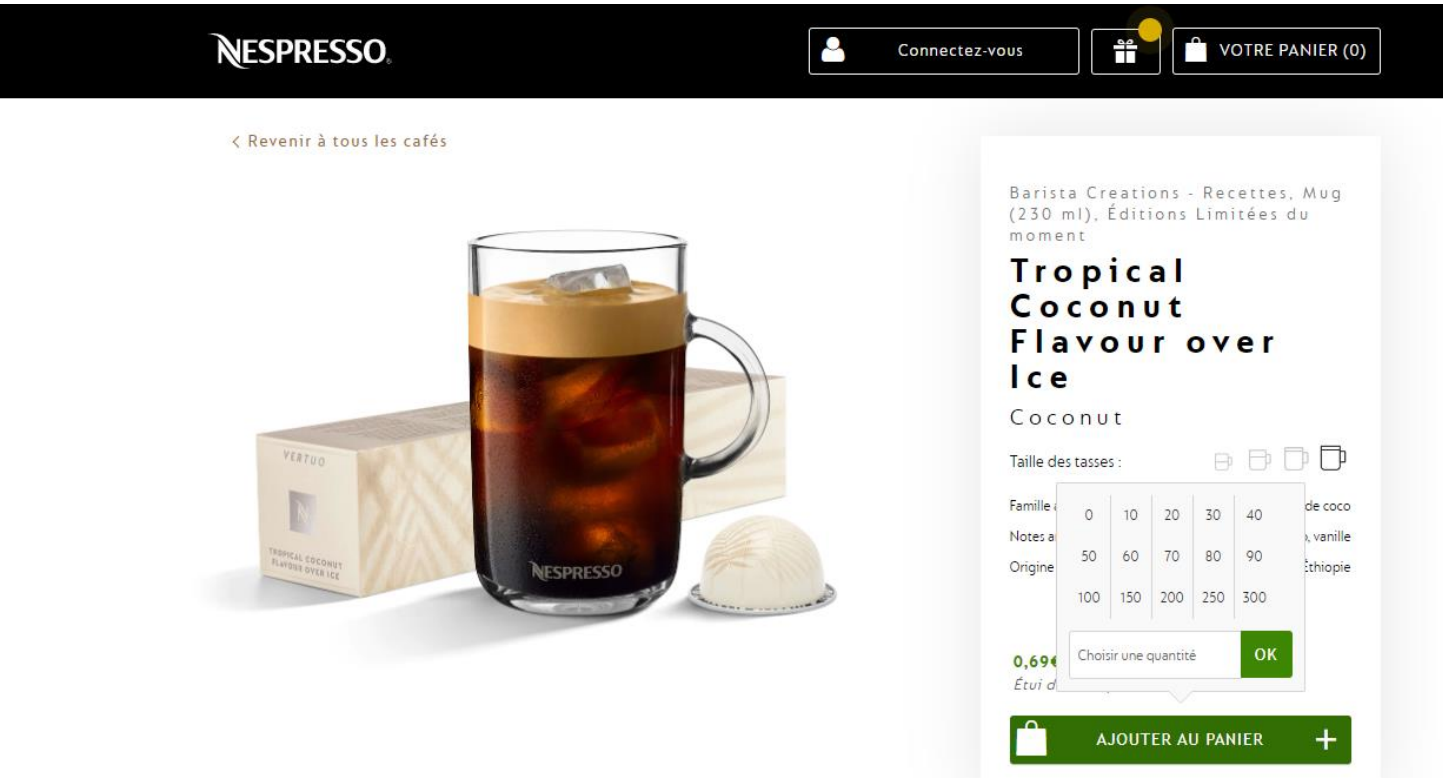


Figure 51: détails d'un produit

L'option de cotation étant activée pour le canal b2c du marché français, une requête « **POST /fr/b2c/anonymous/quoted** » est envoyée à ecapi-carts et un devis est établi pour le client anonyme.

Le client consulte son panier et les prix affichés cette fois-ci sont exacts. Il peut continuer à naviguer pour trouver d'autres produits ou passer directement à la caisse.

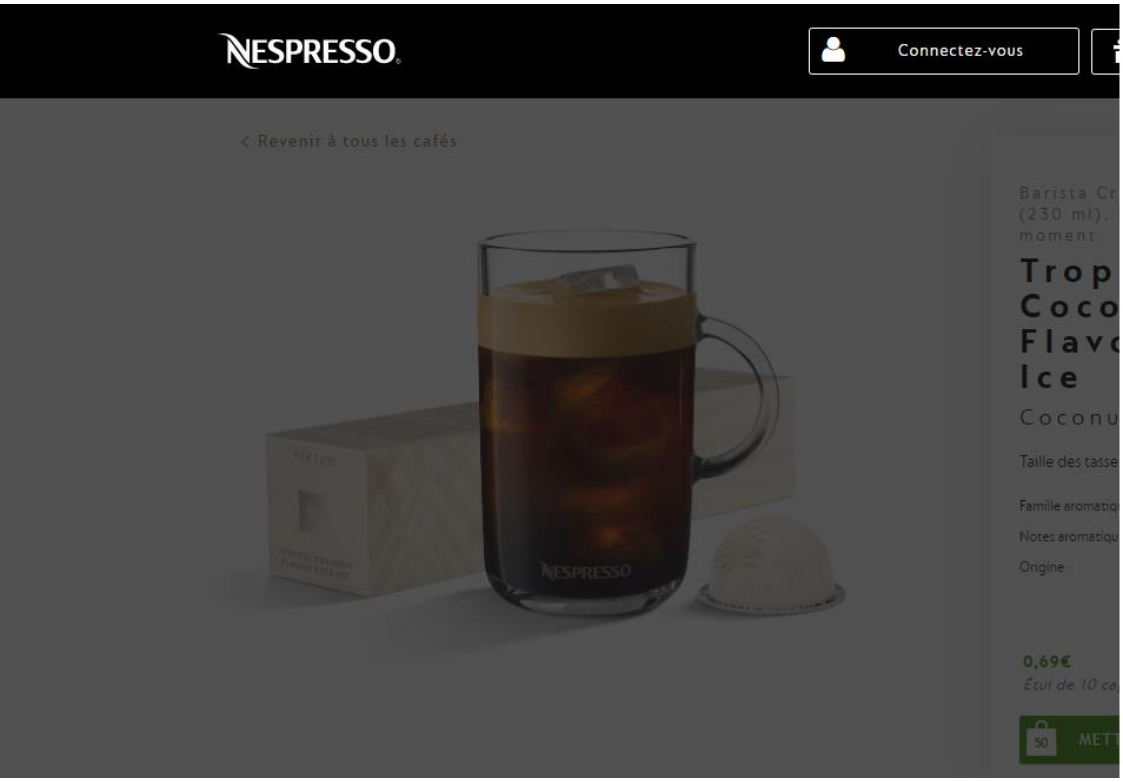


Figure 52: cotation anonyme

Le client souhaite accéder à son compte, alors qu'il a déjà ajouté des articles à son panier en mode anonyme.

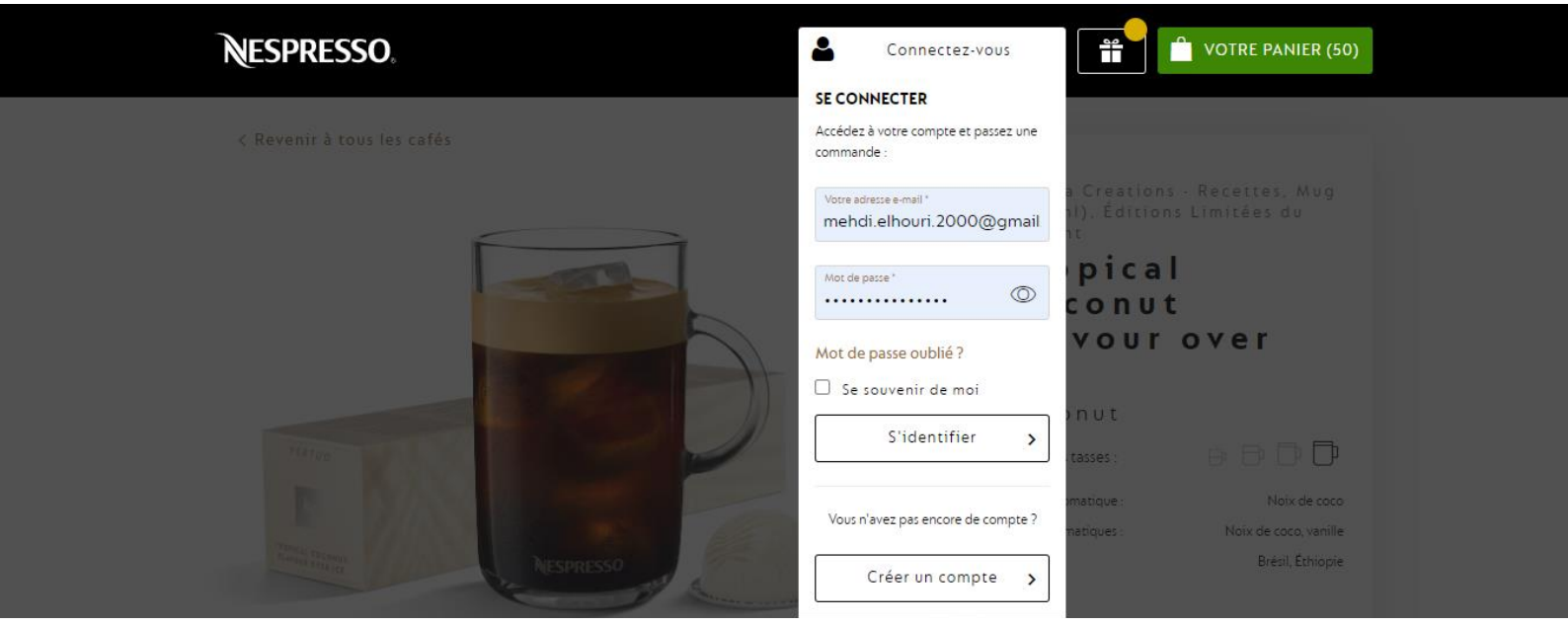
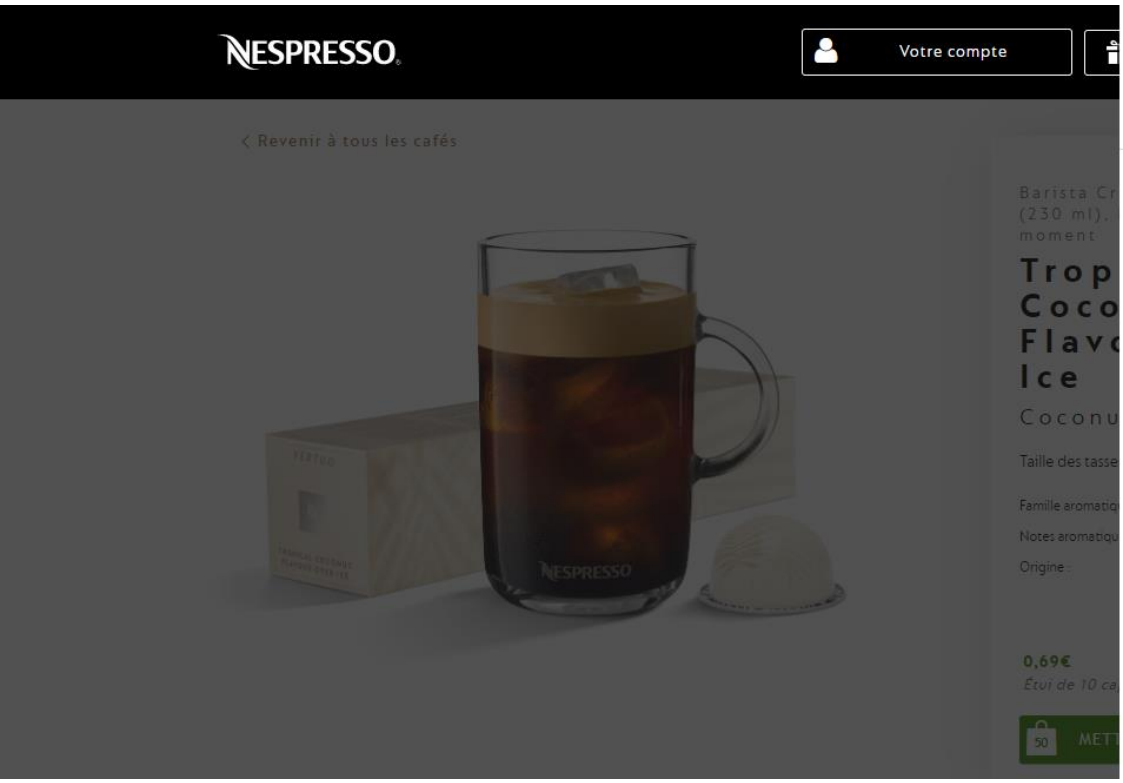


Figure 53: authentification

Cela ne pose pas de problème, le client accède à son compte et le panier qu'il a rempli en mode anonyme est fusionné avec le panier enregistré sur son compte. De plus, son panier est coté et il bénéficie d'une réduction maintenant qu'il est authentifié.

L'option de cotation étant activée pour le canal b2c du marché français, une requête « **POST /fr/b2c/me/quoted** » a été envoyée à ecapi-carts.



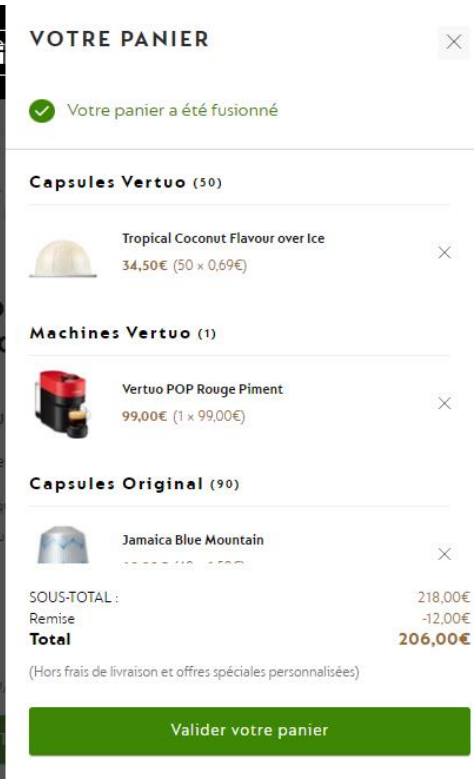


Figure 54: fusion du panier d'achat

Maintenant qu'il est authentifié, le client peut mettre à jour son panier et les modifications sont persistées.

L'option de cotation étant activée pour le canal b2c du marché français, une requête « **PUT /fr/b2c/me/quoted** » est envoyée à ecapi-carts et un nouveau devis est établi.

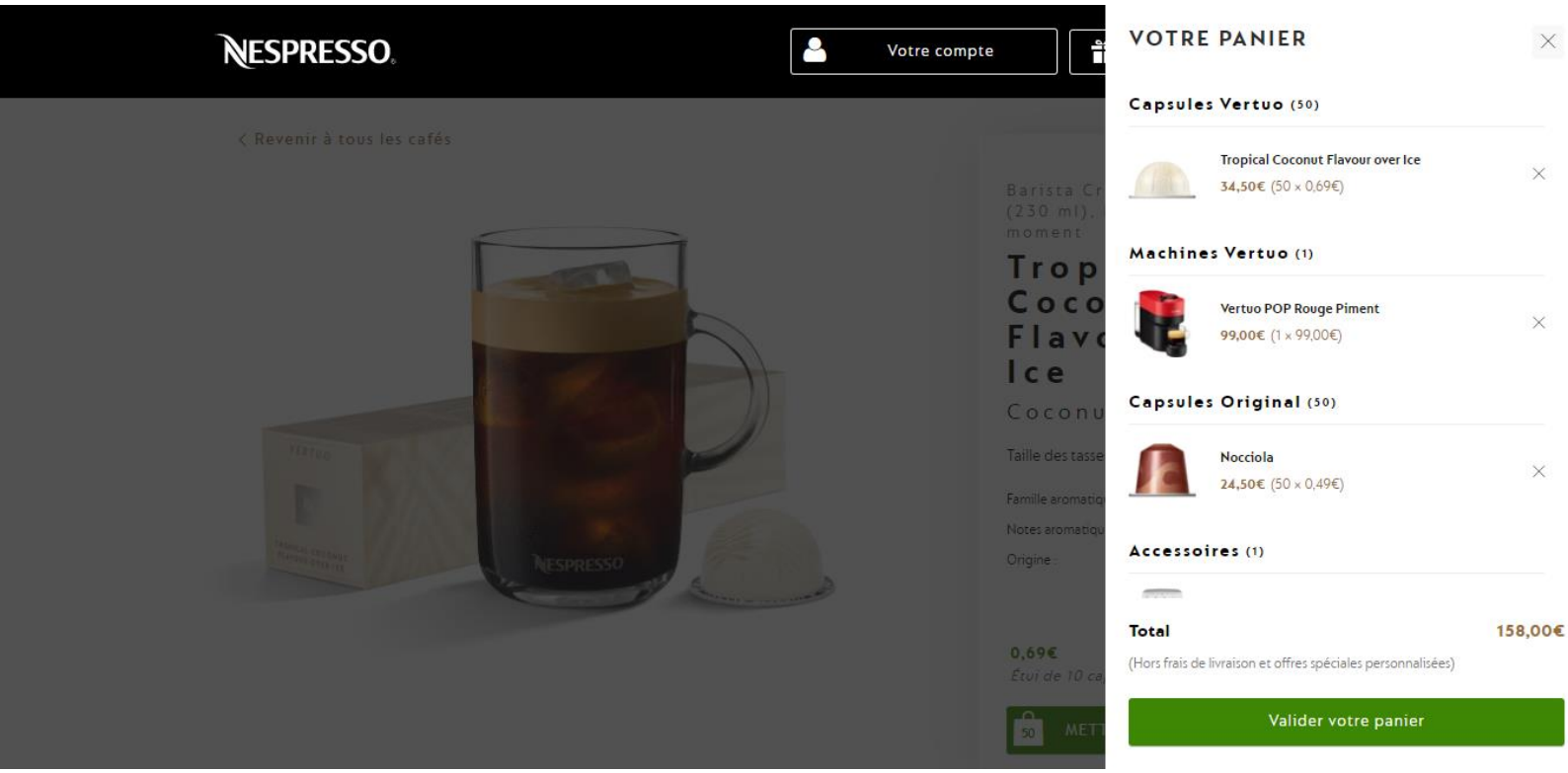


Figure 55: mise à jour du panier d'achat

IV. Documentation de l'API

Un contrat Swagger, également connu sous le nom de documentation API, est un document qui définit la structure, les endpoints, les paramètres, les réponses et d'autres détails d'une API conformément à la norme OpenAPI.

Les contrats Swagger sont rédigés au format YAML ou JSON. Ils sont divisés en plusieurs sections, y compris :

- Section « **Info** » : fournit des informations générales sur l'API, telles que son titre, sa version et sa description ;
- Section « **Paths** » : définit les endpoints de l'API. Chaque endpoint est décrit par sa méthode (GET, POST, PUT, DELETE), son chemin et ses paramètres ;
- Section « **Models** » : définit les types utilisés par l'API. Ces types peuvent être utilisés pour définir la structure des corps des requêtes et des réponses ;
- Section « **Security** » : Cette section définit les exigences de sécurité pour l'API.

Ci-après le contrat Swagger pour ecapi-carts.

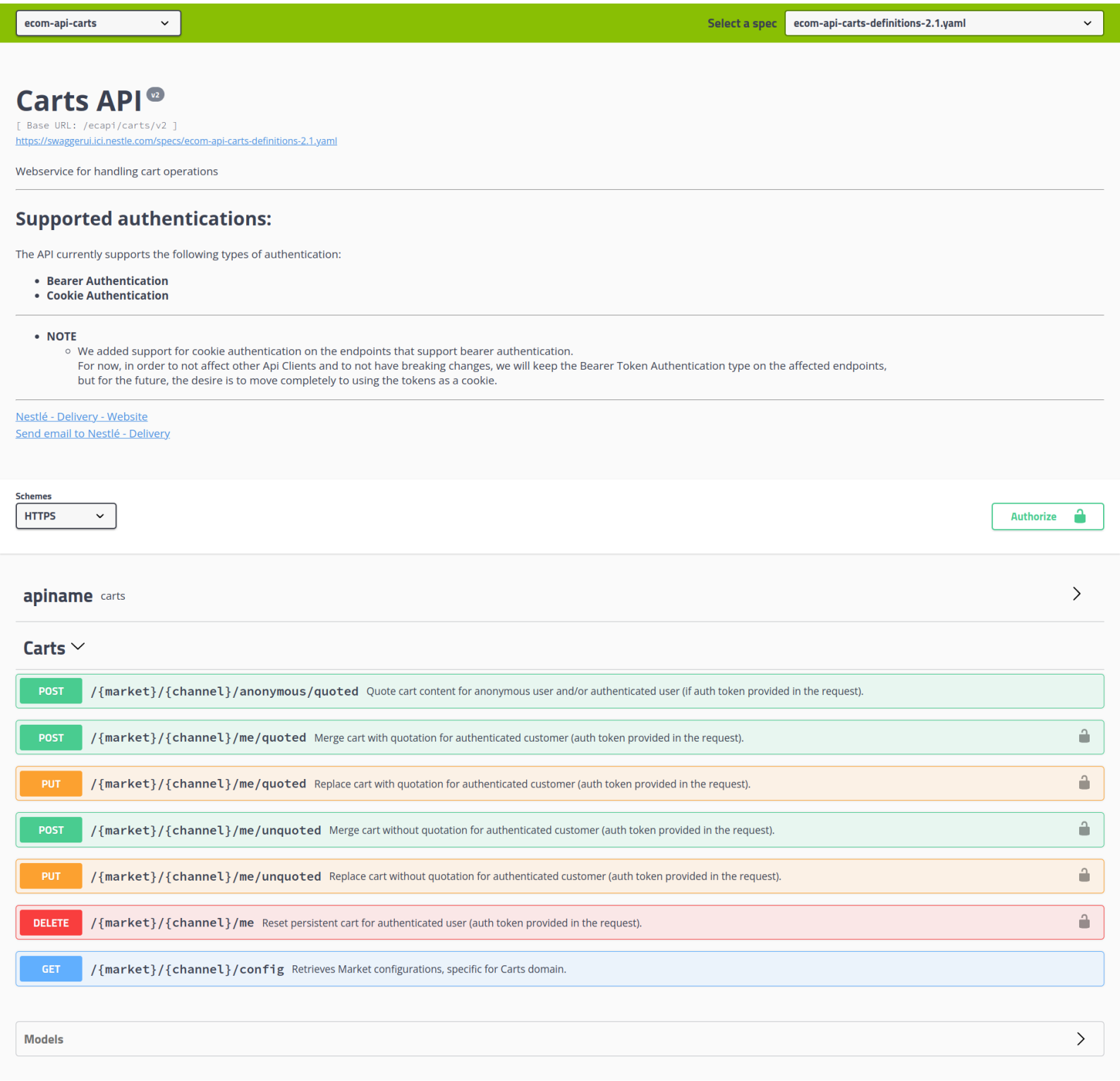


Figure 56: contrat Swagger pour ecapi-carts

Il est également possible de fournir une description détaillée des paramètres de chaque endpoint et de ses différentes réponses possibles. Ci-dessous un exemple pour le premier endpoint.

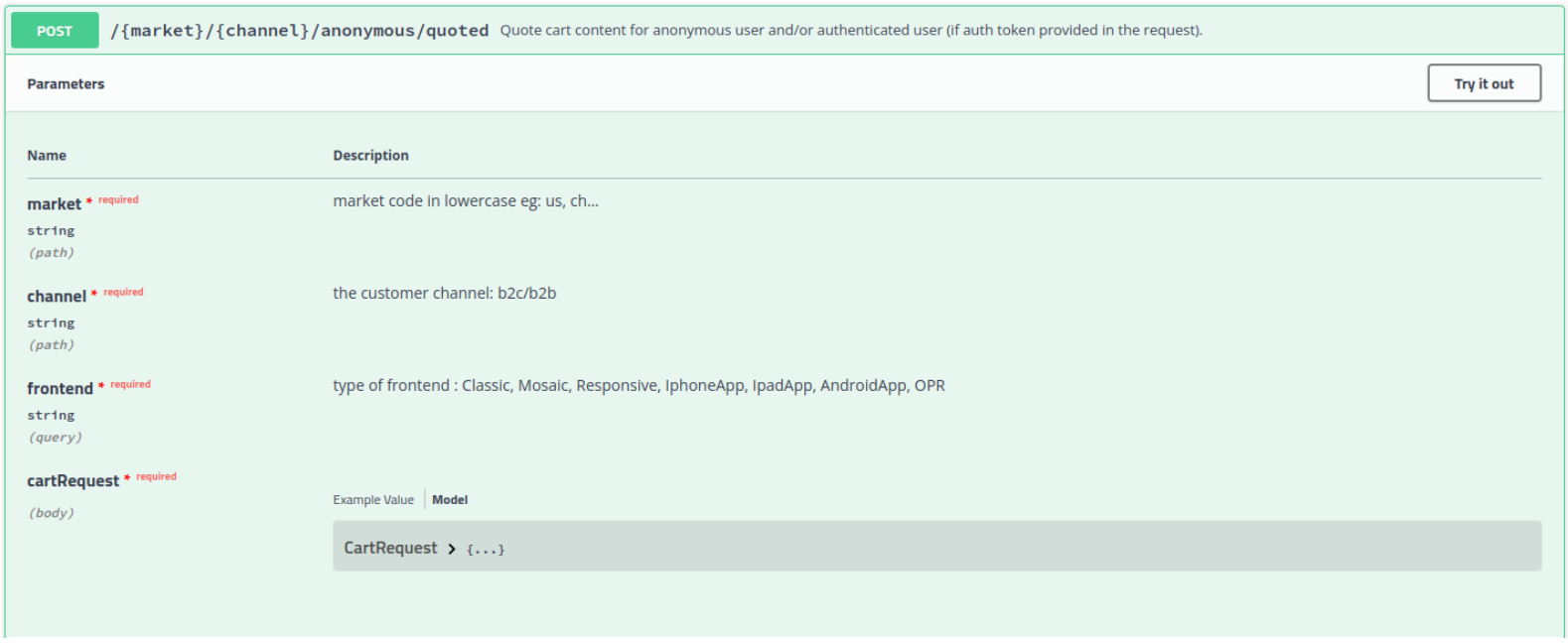


Figure 58: description détaillée des paramètres du premier endpoint

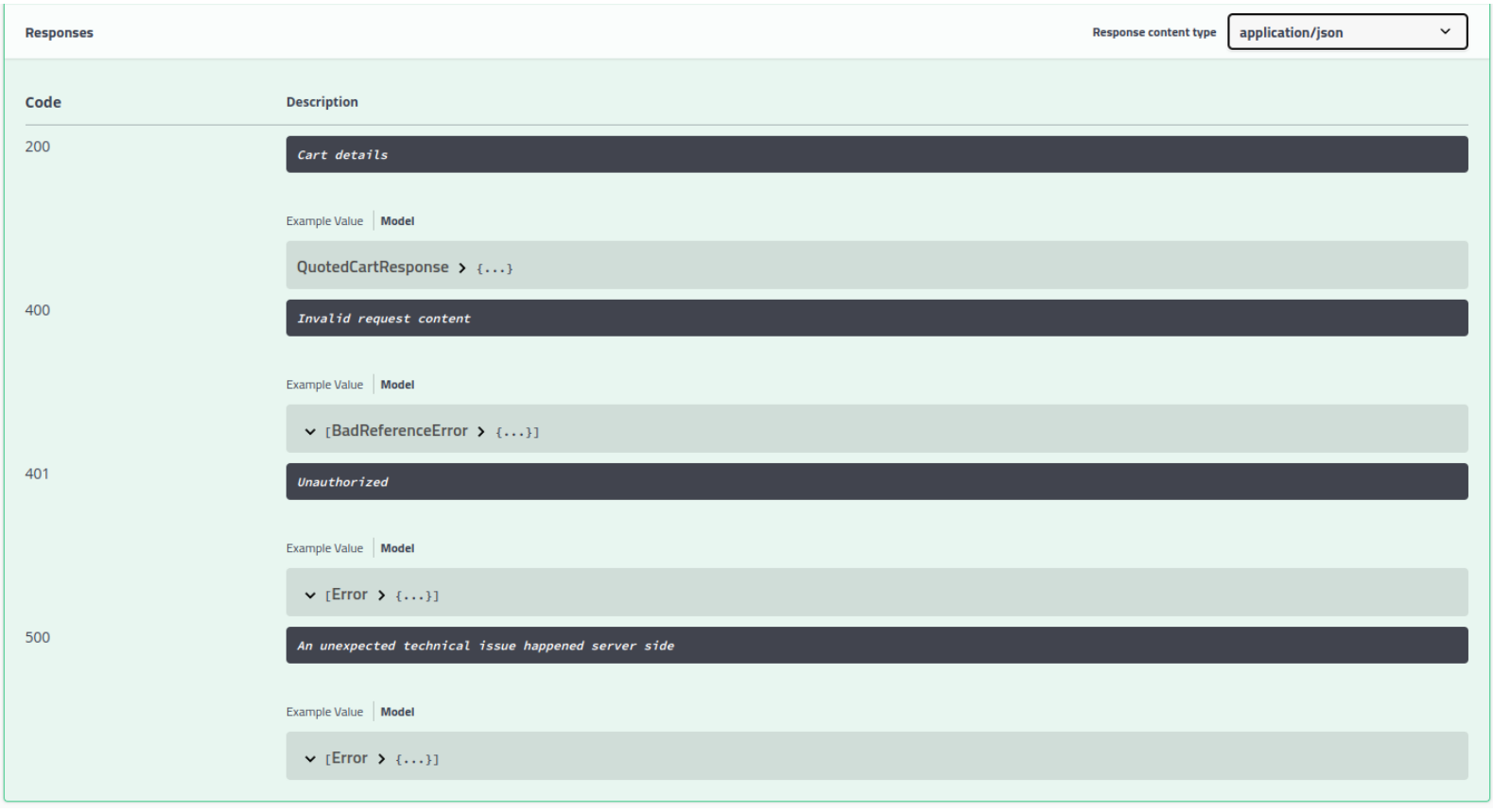


Figure 57: description détaillée des différentes réponses du premier endpoint

V. Monitoring

La mesure, la surveillance et les alertes sont des concepts interconnectés qui, ensemble, constituent la base d'un système de monitoring. Ils permettent d'évaluer l'état d'un système et de comprendre les conséquences des modifications apportées. Lorsque les mesures s'écartent des plages prédéfinies, ces systèmes peuvent envoyer des notifications, incitant les responsables à prendre les mesures nécessaires. En outre, ils facilitent le processus d'investigation des causes potentielles en fournissant des informations précieuses.

Dans cette optique, le projet eCAPI Split fournit des métriques qui renvoient un ensemble d'informations sur les appels aux services exposés par le module ecapi-carts.

A titre d'exemple, les graphiques suivants permettent de surveiller en temps réel la performance du service de cotation en suivant son taux d'erreur (nombre de tentatives, nombre de tentatives réussies et nombre de tentatives échouées).

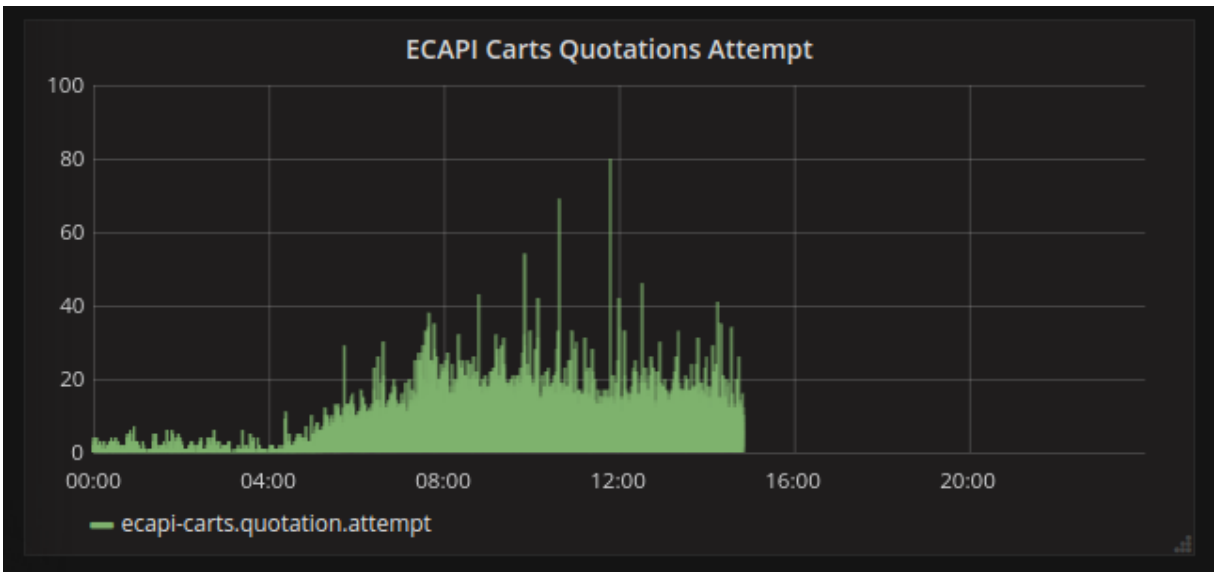


Figure 59: nombre d'appels au service de cotation, affiché dans Grafana

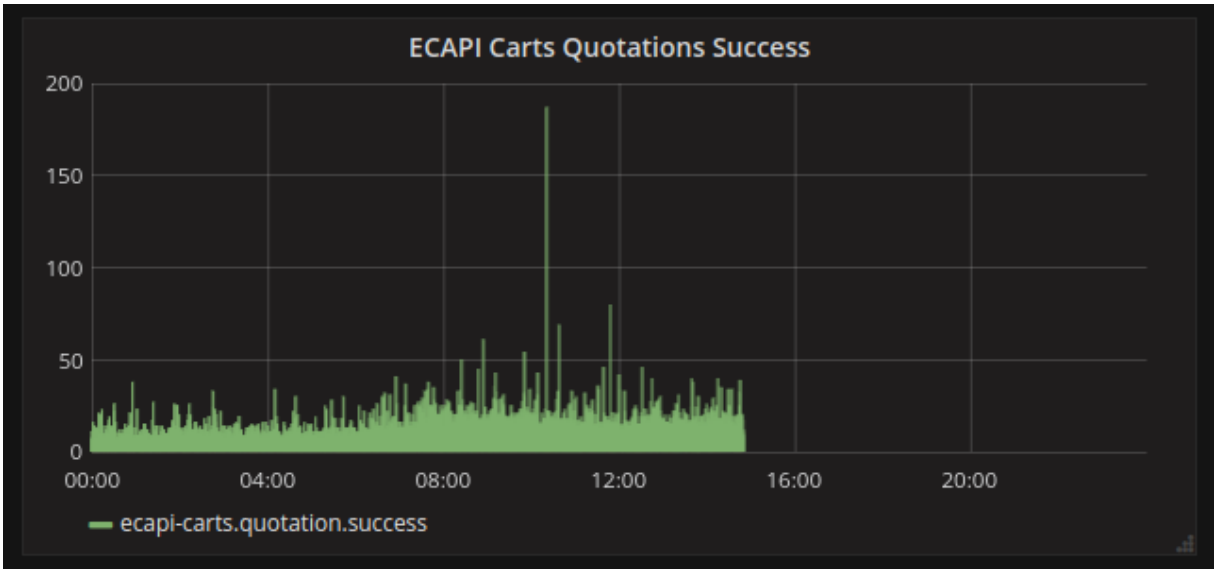


Figure 60: nombre d'appels réussis au service de cotation, affiché dans Grafana

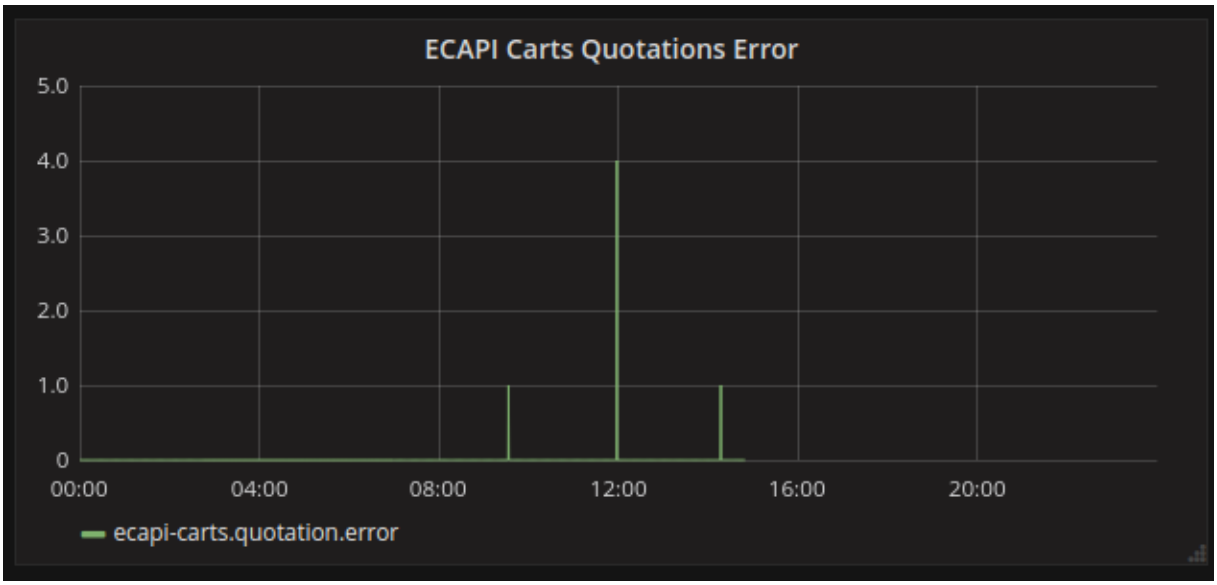


Figure 61: nombre d'appels échoués au service de cotation, affiché dans Grafana

Les endpoints impliqués dans le service de cotation sont :

- **"POST** /{market}/{channel}/anonymous/quoted**"**
- **"POST** /{market}/{channel}/me/quoted**"**
- **"PUT** /{market}/{channel}/me/quoted**"**

Dans l'ensemble, ce chapitre a permis de comprendre l'architecture, la structure du répertoire du code source, les scénarios d'utilisation, la documentation et les aspects liés au monitoring du module ecapi-carts, fournissant ainsi une vue d'ensemble complète de la réalisation du projet eCAPI Cart Split.

Conclusion

Conclusion

La mission de ce stage d'application était axée sur ma contribution à la maintenance adaptative de la plateforme e-commerce de la marque Nespresso, notamment le module « Panier d'achat », visant à répondre aux besoins des clients particuliers (B2C) comme à ceux des clients professionnels (B2B). Le projet « eCAPI Cart Split » fait partie de la collection des sous projets « eCAPI Split » du projet Nespresso adopté par SQLI, et met en place tous les processus garantissant une bonne qualité logicielle.

La phase suivante de l'initiative "eCAPI Split" concerne le sous-projet "eCAPI Customer Split", qui vise à transférer le service web "eCAPI Customer" vers un microservice autonome. Ce processus de migration sera mené sur la base d'une méthodologie similaire à celle suivie pour le projet "eCAPI Cart Split". Il en va de même pour tous les autres domaines (services web) de la plateforme NC2. La transition vers l'architecture microservices cible "eCAPI Split" sera alors progressive et exhaustive.

Finalement, je me sens privilégiée d'avoir eu la chance de contribuer à un projet aussi important et d'être membre des équipes SQLI et Nespresso. Tout au long de mon stage, j'ai eu l'occasion d'affiner les compétences que j'ai acquises au cours de mes études universitaires, d'élargir mon expertise technique et d'explorer de nouveaux outils fascinants.

Par ailleurs, sur le plan personnel, j'ai eu le privilège de rencontrer plusieurs cadres au cours de mon stage, à qui je tiens à exprimer ma gratitude. Ces interactions m'ont permis d'engager des discussions instructives sur divers sujets, améliorant ainsi mes compétences en matière de communication et renforçant ma capacité à collaborer efficacement au sein d'une équipe.

Références

[1] SQLI, [En ligne]. Available: sqli.com/ma-fr.

[2] SQLI, [En ligne]. Available: sqli.com/ma-fr/groupe/groupe-sqli-novateur-et-generateur-dexperience-depuis-1990.

[3] Wikipédia, «SQLI,» [En ligne]. Available: fr.wikipedia.org/wiki/SQLI.

[4] Wikipédia, «Nespresso,» [En ligne]. Available: fr.wikipedia.org/wiki/Nespresso.

[5] Teamhood, «SCRUM,» [En ligne]. Available: teamhood.com/agile/scrum-cheat-sheet.

[6] Oracle, «Java 8,» [En ligne]. Available: docs.oracle.com/javase/8/docs/technotes/guides/language/index.html.

[7] Wikipédia, «Spring,» [En ligne]. Available: fr.wikipedia.org/wiki/Spring_(framework).

[8] Rollbar, «Spring Boot,» [En ligne]. Available: rollbar.com/blog/spring-vs-spring-boot-whats-the-difference/.

[9] J.-M. DOUDOUX, «JUnit,» [En ligne]. Available: jmdoudoux.fr/java/dej/chap-junit.htm.

[10] DOCDOKU, «Mockito,» [En ligne]. Available: docdoku.com/blog/2015/03/23/tests-unitaires-avec-mockito/.

[11] Wikipédia, «Cassandra,» [En ligne]. Available: fr.wikipedia.org/wiki/Cassandra_(base_de_données).

[12] Wikipédia, «IntelliJ IDEA,» [En ligne]. Available: en.wikipedia.org/wiki/IntelliJ_IDEA.

[13] Wikipédia, «Git,» [En ligne]. Available: en.wikipedia.org/wiki/Git.

[14] Wikipédia, «Bitbucket,» [En ligne]. Available: en.wikipedia.org/wiki/Bitbucket.

[15] AWS, «Docker,» [En ligne]. Available: aws.amazon.com/fr/docker/.

[16] Edureka, «What is Jenkins,» [En ligne]. Available: edureka.co/blog/what-is-jenkins/.

[17] Kubernetes, «Kubernetes,» [En ligne]. Available: kubernetes.io/fr.

[18] Wikipédia, «Swagger,» [En ligne]. Available: fr.wikipedia.org/wiki/Swagger_(logiciel).

[19] Product Plan, «Jira,» [En ligne]. Available: productplan.com/glossary/jira/.

[20] Indeed, «What is Confluence,» [En ligne]. Available: uk.indeed.com/career-advice/career-development/what-is-confluence.

[21] Wikipédia, «Prometheus,» [En ligne]. Available: fr.wikipedia.org/wiki/Prometheus_(logiciel).

[22] Wikipédia, «Grafana,» [En ligne]. Available: en.wikipedia.org/wiki/Grafana.

[23] Wikipédia, «SonarQube,» [En ligne]. Available: fr.wikipedia.org/wiki/SonarQube.

[24] SonarQube Docs, «SonarQube,» [En ligne]. Available: docs.sonarqube.org/latest/.

[25] Microsoft, «An introduction to domain driven design,» [En ligne]. Available: learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/best-practice-an-introduction-to-domain-driven-design.

[26] M. Schaefer, «Onion architecture explained,» [En ligne]. Available: marcoatschaefer.medium.com/onion-architecture-explained-building-maintainable-software-54996ff8e464.